UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Sincronización de directorios locales de Linux con directorios HDFS de Hadoop

Sergio Yunta López
Tutora: Rosa Mª Carro Salas

Mayo 2018

Sincronización de directorios locales de Linux con directorios HDFS de Hadoop

AUTOR: Sergio Yunta López
TUTORA: Rosa Mª Carro Salas

Grupo de Herramientas Interactivas Avanzadas
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2018

Agradecimientos

En primer lugar agradecer a mi familia toda la fe depositada en mí, pues en numerosas ocasiones ellos han creído más que yo y nunca han dudado en que lo conseguiría. Gracias a mi madre, Carmen, por aguantar mis innumerables tardes de mal humor en casa, a mi hermano Luis Ángel, pues aún en la distancia, sé que está ahí, y en especial, a mi hermano Roberto, quién nunca me ha dejado que tire la toalla, ya que a pesar de ser muy estricto conmigo y exigirme siempre con todo, en ningún momento me ha faltado su apoyo en los momentos más difíciles. Además, agradecer la ayuda ofrecida por mi cuñada Eva, quién siempre está dispuesta a "echarme una mano".

A mi tutora Rosa Carro, por interesarse por este trabajo y darme la posibilidad de poder realizarlo, proporcionándome consejos y apoyo con los problemas que han ido surgiendo.

A todos los profesores que me han impartido clase durante estos años, gracias a ellos me he ido formando poco a poco y lo más importante, me han enseñado a pensar como un informático.

También agradecer a mis amigos del pueblo, los cuales llevan a mi lado desde la infancia y aunque es cierto que existen muchos piques entre nosotros, siempre sabemos cómo levantarnos el ánimo los unos a los otros, ayudándonos a crecer como personas. Sin ellos todo hubiera sido más difícil, especialmente sin mi primo y amigo Josete, él jamás me ha juzgado y siempre he podido contar con su ayuda.

Finalmente, no podría olvidarme de mis amigos y compañeros de la universidad. Agradecerles a todos ellos los buenos ratos pasados en los laboratorios y en la biblioteca, pero también los malos, en los cuales siempre conseguían que nada fuera tan gris como parecía. Gracias a todos "Grapastagorianos", pero sobre todo gracias a cuatro personas que considero como hermanos y que se han convertido en una parte muy importante de mi vida, Rone, Tito, Iago y Lebron. Sin ellos estoy seguro de que no podría haberlo logrado.

¡GRACIAS A TODOS!

Resumen

En los últimos años se han producido grandes avances tecnológicos y científicos en todos los ámbitos de la sociedad. En el ámbito de la informática se han asumido nuevos retos a la hora de almacenar y manejar información de todo tipo. El crecimiento que ha experimentado "Internet" en un breve periodo de tiempo, así como el incremento exponencial de información disponible en distintas fuentes y con distintos formatos, es decir, la llegada del "Big Data", han hecho que la manipulación de datos tal y como la conocíamos haya cambiado totalmente.

Actualmente se trabaja con mucha más información digital que en un principio cuándo surgió Internet. Las distintas formas que existían de almacenar datos se han ido informatizando con el paso de los años, de la misma manera que también se han digitalizado los propios datos almacenados, para que no ocupen espacio físico real. Esto ha ocasionado que cada vez existan más ficheros digitales, en vez de documentos impresos en papel, con información relevante que es necesario conservar. Para favorecer y ayudar en el almacenamiento y administración de toda esta información se ha creado "Apache Hadoop". Este *framework* permite manipular grandes cantidades de datos mediante clústers, a través de un sistema de ficheros distribuido, el cual permite que el fichero de datos no se guarde en una única máquina, dejando a las aplicaciones ejecutarse en varios servidores.

En este trabajo se ha diseñado y desarrollado una herramienta software capaz de ayudar a Hadoop a tener actualizados los ficheros que almacena, para poder consultar la versión más reciente posible de los datos, y que a la vez permite al usuario realizar distintas acciones sobre los archivos existentes de manera sencilla.

Con la herramienta desarrollada (RsyncHdfs) se consigue sincronizar archivos y directorios locales de Linux con el sistema de almacenamiento distribuido de Hadoop. De la misma forma, también se puede sincronizar el entorno de Linux a partir de elementos existentes solo en Hadoop.

Habiendo terminado la última versión del prototipo, el potencial del proyecto se ha considerado cuando se han realizado diversas pruebas en un entorno real. Durante la realización de las mismas, se ha podido comprobar la usabilidad de la herramienta creada, dando por cumplidos los objetivos propuestos en las fases iniciales del proyecto.

Palabras clave

Big Data, Apache Hadoop, *framework*, clúster, herramienta software, sincronizar archivos, almacenamiento distribuido.

Abstract

In recent years there have been lot of technological and scientific advances in all society areas. Particularly, on informatics area lot of challenges have been assumed for storing and managing the information. The significant growth of "Internet" on a brief period of time, and the new appearance of "Big Data" term, have made that data managing has changed since then.

Actually, digital information volume has been increasing and how it is processing has changed since the beginning. Lots of data has been digitalized for minimized physical storage. Consequently, virtual storage has been increased with relevant information that is necessary to preserve. In that sense, for encouraging and helping in the storage of all of this information, "Apache Hadoop" has been created. This framework allows to manage large amounts of data via clusters, with a distributed file system (DFS) that save the file on more than one server for running the programs on several servers.

On this TFG, we have thought on a software tool capable of supporting Hadoop to keep updated all files storage on it, for consulting the newest one, and allowing the user perform different actions on this existing files.

With RsyncHdfs it is possible synchronize files and local Linux directories with HDFS, at the same way, it is also possible synchronize Linux environment from existing elements only on Hadoop.

With the last prototype version finished, it has been run out on and tested on a real environment. During the testing, durability and usability of the created tool has been checked, and the results have concluded the objectives to be achieved on the initial project phases.

Keywords

Big Data, Apache Hadoop, framework, cluster, software tool, sinchronize files, distributed storage.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

1 Introducción

Debido a la gran cantidad de información con la que se trabaja hoy en día, es necesario disponer de herramientas para poder almacenar y procesar dicha información de forma distribuida, pues a menudo con un solo ordenador no es suficiente. Es así como surge *Apache Hadoop,* un *framework* con el cual se pueden manipular grandes volúmenes de datos a través de clústers.

El sistema de almacenamiento de Hadoop o sistema HDFS consiste en un sistema de ficheros distribuido que permite que los ficheros de datos no se guarden en una única máquina, permitiendo a las aplicaciones ejecutarse en varios servidores.

Para poder mantener los datos de manera consistente, es necesario actualizar multitud de ficheros y añadir o eliminar numerosos archivos, comprobando que no se está eliminando información que se desea guardar. Por esta razón, es necesario tener sincronizados constantemente los datos que se guardan en Hadoop con los datos que estamos manipulando de manera local en nuestro equipo.

Este Trabajo de Fin de Grado, tiene como propósito el desarrollo de una herramienta que permita sincronizar los datos almacenados en un sistema Unix de Linux con la información almacenada en el sistema HDFS. En este capítulo se detalla una breve descripción de los motivos que han llevado a la realización de este proyecto, así como sus objetivos principales.

1.1 Motivación

Los seres humanos estamos creando y almacenando información constantemente y cada vez más en cantidades astronómicas. Se podría decir que si todos los bits y bytes de datos del último año fueran guardados en CD’s o DVD’s y se apilaran todos ellos, se generaría una gran torre desde la Tierra hasta la Luna. En este contexto, poder mantener la información actualizada y ordenada se ha convertido en algo imprescindible.

Esta contribución a la acumulación masiva de datos se puede observar en diversas industrias. Un ejemplo son las grandes empresas o compañías multinacionales, las cuales mantienen grandes cantidades de datos transaccionales, reuniendo información acerca de sus clientes, proveedores, operaciones, etc.

Gracias al sistema de almacenamiento de Hadoop, las empresas pueden guardar los datos mencionados anteriormente mediante ficheros, los cuales serán manejados por Hadoop y acumulados en Hive como si se tratara de una base de datos. Como es obvio, para poder utilizar esta gran cantidad de datos de manera satisfactoria, se necesita disponer de la información más actual y poder sincronizarla en caso de que se realicen cambios sobre la misma, razón que convierte en interesante la realización de este Trabajo de Fin de Grado sobre la creación de un sincronizador de directorios locales de Linux con directorios HDFS de Hadoop.

1.2 Objetivos

El objetivo de este proyecto es diseñar e implementar una herramienta que permita sincronizar archivos y directorios locales de Linux con archivos y directorios que se encuentran en el sistema de almacenamiento distribuido de Hadoop. El funcionamiento de esta herramienta (que se llamará "RsyncHdfs") será similar al de "Rsync" existente en sistemas de tipo Unix y Windows, basado en la transferencia y actualización de archivos y directorios entre una ubicación origen con una ubicación destino.

Rsync permite la sincronización entre dos ubicaciones de una misma máquina o dos máquinas que se encuentren dentro de una misma red con el mismo sistema operativo. Lo que se pretende conseguir con RsyncHdfs es sincronizar archivos y directorios locales de Linux con archivos y directorios que se encuentran en el sistema de almacenamiento distribuido de Hadoop. En concreto, el funcionamiento de RsyncHdfs será:

1. Mantendrá los permisos y modos de los archivos que se sincronicen dependiendo de las funcionalidades de uso que se especifiquen.

2. Por defecto, si no se indica ninguna funcionalidad, se copiarán archivos de la ubicación origen en la destino, se eliminarán archivos que existan en destino y no se hallen en origen, y se actualizarán los existentes si se encuentran en ambas ubicaciones, siempre y cuando sean más actuales los de origen que los de destino.

3. Permitirá al usuario realizar diferentes tipos de sincronización, como copiado de archivos nuevos o actualización de los existentes, obedeciendo a las diferentes opciones que utilicen, de entre las que posee el sincronizador.

4. Si se añade alguna funcionalidad a la ejecución, mediante algún parámetro existente optativo, es decir, cuyo uso no es necesario para que el sincronizador realice su funcionamiento por defecto, el sincronizador realizará exactamente las órdenes indicadas con los archivos a sincronizar.

5. Si en algún momento de la ejecución, RsyncHdfs falla por cualquier motivo, no se perderá ningún tipo de archivo y/o información que estos contengan. Es decir, en caso de error, la ubicación de origen ha de quedar tal y como estaba en el momento posterior a la ejecución.

1.3 Estructura del documento

- En el capítulo 2 se describe la actualidad de los dos entornos utilizados en la sincronización y se analizan otro tipo de sincronizadores, con el fin de poner al lector al corriente de las herramientas similares existentes para otra clase de sistemas.

- El capítulo 3 especifica el alcance de la aplicación, la metodología aplicada en el desarrollo de este proyecto y analiza la funcionalidad a la que ha de dar soporte el sincronizador RsyncHdfs.

- En el capítulo 4, se detalla el análisis del proyecto, incluyendo los distintos requisitos funcionales y no funcionales que ha de satisfacer el programa.

- En el capítulo 5 se describe el diseño del sincronizador.

- El capítulo 6 se explica el desarrollo de RsyncHdfs.

- A lo largo del capítulo 7 se muestran las pruebas realizadas durante y tras el desarrollo de la aplicación, y se presentan el resultado de las mismas.

- Por último, en el capítulo 8 se recogen las conclusiones del trabajo realizado, así como las posibles mejoras y/o evolución futura del mismo.

2 Estado del Arte

En este capítulo se explica una breve historia y se describen la organización y las ventajas tanto de los sistemas Linux como del sistema de almacenamiento Hadoop. Por otro lado, se realiza un estudio de tecnologías existentes actualmente similares a RsyncHdfs, entre las cuales se encuentra "Rsync", herramienta en cuyo tipo de funcionamiento se basa este proyecto.

2.1 Linux

Linux fue creado originalmente por "Linus Torvald" en la Universidad de Helsinki (Finlandia), pero ha continuado y continúa su desarrollo a través de Internet, con la ayuda de otros numerosos programadores, ya que se trata de un sistema operativo de código abierto.

Torvald basó el desarrollo de Linux en Minix, un pequeño sistema Unix desarrollado por "Andy Tannenbaum", proponiéndose crear, como él mismo dijo, un –*"mejor Minix que el Minix"*–. El 5 de Octubre de 1991, Torvald anunció su primera versión oficial de Linux, la versión 0.02. Su antecesora, la versión 0.01, nunca fue publicada al no tratarse de una versión ejecutable, ya que estaba escrita en lenguaje ensamblador y asumía que se tenía acceso a un sistema Minix para su compilación.

2.1.1 Estructura de directorios Linux

El estándar utilizado para organizar la información en Linux se denomina FHS (*Filesystem Hierarchy Standard*) y, se encarga de organizar la información de forma jerárquica. La Figura 2.1 muestra un resumen de la estructura de directorios y archivos en Linux.

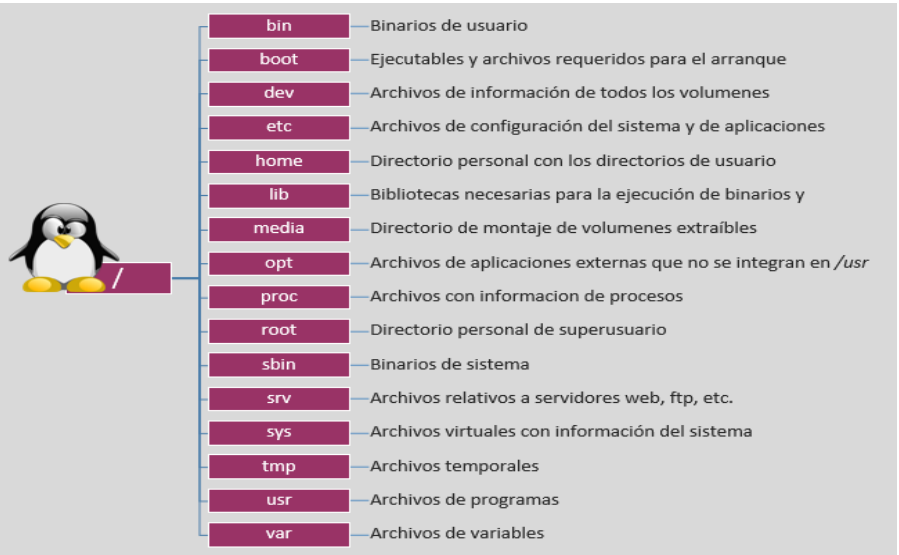

Figura 2.1: Estructura de directorios Linux

Los directorios principales son:

- Directorio raíz (/): de él nacen el resto de directorios, independientemente si están almacenados en disco o en unidades externas. Cualquier dirección de archivo o carpeta en Linux comienza por el directorio raíz, seguido de todos los directorios y subdirectorios que lo contienen, separados cada uno de ellos por "/".

- /bin: se trata de un directorio estático donde se almacenan todos los binarios necesarios para garantizar las funciones básicas a nivel de usuario. Solo almacena los ejecutables de usuario, ya que los binarios imprescindibles para tareas administrativas gestionadas por el usuario *root* (usuario administrador) del sistema se encuentran en el directorio */sbin*. Incluye asimismo los binarios que permiten la ejecución de varias utilidades estándar de la terminal de Linux, concretamente *cat, cd, cp, echo, grep, gzip, kill, ls, mv, rm, ping, su, ps, tar y vi.*

- /boot: es un directorio estático e incluye todos los ejecutables y archivos que son necesarios en el proceso de arranque del sistema. Es también dónde se encuentra el gestor de arranque GRUB.

- /dev: incluye todos los dispositivos de almacenamiento conectados al sistema en forma de archivos, es decir, cualquier disco duro, partición, memoria USB o CDROM conectado y que el sistema pueda entender como un volumen lógico de almacenamiento.

- /etc: es el encargado de recopilar los archivos de configuración tanto a nivel de componentes del sistema operativo, como de los programas y aplicaciones instaladas a posteriori.

- /home: directorio de los usuarios estándar, destinado a almacenar todos los archivos del usuario, como fotos, documentos, vídeos, música, etc. Además contiene archivos temporales de aplicaciones ejecutadas en modo usuario, que sirven para guardar las configuraciones de programas.

- /lib: almacena las bibliotecas esenciales que son necesarias para que se puedan ejecutar correctamente todos los binarios que se encuentran en los directorios */bin* y */sbin*, así como los módulos del propio kernel.

- /media: representa el punto de montaje de todos los volúmenes lógicos que se montan temporalmente, como memorias USB u otras particiones de disco.

- /opt: haciendo una analogía con Windows vendría a ser algo como el directorio de "Archivos y Programas", pero en este caso, para todos aquellos archivos de solo lectura que son parte de programas auto-contenidos.

- /proc: contiene información sobre los procesos y aplicaciones que se están ejecutando en un momento determinado en el sistema. Almacena archivos virtuales, por lo que no guarda nada como tal y su contenido es nulo.
- /root: es el directorio /home del superusuario o usuario root.

- /sbin: guarda archivos binarios al igual que /bin, con la diferencia de que estos binarios son los relativos a tareas propias del sistema operativo que solo pueden ser gestionadas por el usuario root.

- /srv: sirve para acumular archivos y directorios relativos a servidores que puedan estar instalados dentro del sistema operativo.

- /sys: al igual que /proc, almacena archivos virtuales que contienen información del *kernel* relativa a eventos del sistema operativo.

- /tmp: en él se depositan archivos temporales de todo tipo, ya sea de elementos del sistema o de diferentes aplicaciones a nivel de usuario.

- /usr: su nombre viene de *User System Resources* y actualmente sirve para almacenar todos los archivos de solo lectura y relativos a las utilidades de usuario, incluyendo todo el software instalado a través de los gestores de paquetes de cada distribución.

- /var: recoge varios archivos con información del sistema, como archivos de logs, bases de datos, información almacenada en la caché, etc.

2.1.2 Ventajas al utilizar Linux

Existen numerosas ventajas fundamentales ofrecidas por Linux. Es robusto, estable y rápido, lo que lo convierte en un gran sistema para servidores y **aplicaciones distribuidas**. Admite cualquier tipo de dispositivo, lo que brinda a dicho software de una gran adaptabilidad sin limitarse como otros sistemas operativos. Para la memoria virtual usa paginación a disco, permitiendo una partición o un archivo, o ambos con la posibilidad de añadir más áreas de intercambio según avanza.

Por otro lado, se trata de un sistema multitarea y multiusuario. Esto permite que varios usuarios accedan a las aplicaciones y recursos del sistema al mismo tiempo, pudiendo además ejecutar varios programas a la vez. También cuenta con *shells* programables e independencia de dispositivos y comunicaciones.

De igual forma, es destacable que dicho sistema es libre, lo cual implica, no solo que es gratuito, sino que puede modificarse y posee una gran cantidad de aplicaciones libres en Internet. Existen diferentes distribuciones con las que podemos trabajar como *RedHat, Debían, Slackware*, etc.

2.2 Hadoop

Big Data pasó en poco tiempo de ser un conjunto de tecnologías innovadoras a convertirse en un mercado, transformándose hoy prácticamente en una industria. El sistema más utilizado en esta industria, para ofrecer capacidades analíticas avanzadas es *Hadoop*, un software de código abierto cuyo desarrollo está coordinado por *Apache Foundation*. Dicho software facilita el almacenamiento de información y permite hacer con rapidez consultas complejas sobre las bases de datos existentes.

El origen de Hadoop se remonta al año 2004, cuando el ingeniero de software, "Doug Cutting", describe en un documento técnicas para manejar grandes volúmenes de datos, dividiéndolos en problemas más pequeños para así poder abordarlos. La procedencia del nombre es mucho menos técnica de lo que se podría esperar, ya que el software recibe el nombre de Hadoop debido a que el hijo de tres años de Cutting llamaba de esta forma a su peluche favorito, razón por la cual el inventor bautizó así a la plataforma. También del peluche de su hijo tomaría su logo, un elefante amarillo.

2.2.1 Arquitectura de Hadoop

Hadoop fue creado a partir del *Google File System* (GFS). Se caracteriza por ser un sistema de ficheros distribuido (HDFS), optimizado para grandes flujos y por trabajar con ficheros grandes en sus lecturas y escrituras. Su diseño reduce la E/S en la red.

A continuación se explican los dos elementos más importantes que forman un clúster Hadoop y se muestran en la Figura 2.2.1.

- *NameNode*: sólo hay uno en el clúster y sirve para regular el acceso a los ficheros por parte de los clientes. Mantiene en memoria los metadatos del sistema de ficheros y un registro de los bloques de fichero que posee cada *DataNode*.

- *DataNode*: es el responsable de leer y escribir las peticiones de los clientes. Los ficheros que contienen están formados por bloques, que se encuentran replicados en diferentes nodos.

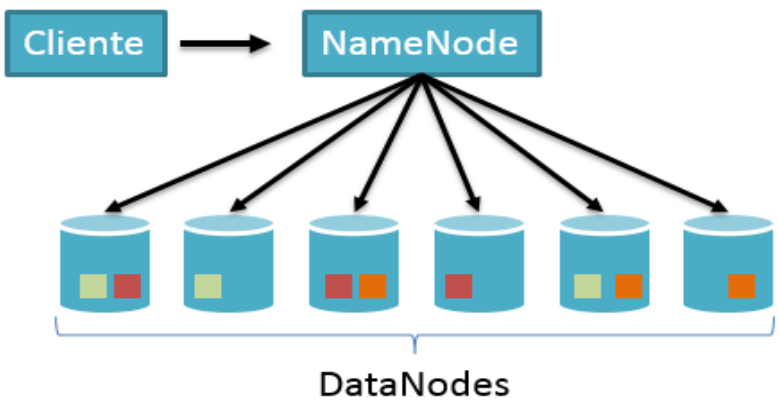

Figura 2.2.1: Arquitectura Hadoop

2.2.2 Ventajas al utilizar Hadoop

La arquitectura de HDFS permite llevar a cabo un análisis eficaz de grandes datos no estructurados, añadiéndoles un valor para poder tomar decisiones estratégicas, mejorar procesos de producción, ahorrar costes o extraer conclusiones científicas. Esto es posible gracias a su tecnología escalable, velocidad, flexibilidad, almacenamiento y tolerancia ante fallos. Seguidamente se explican con más detalle estos cinco puntos fuertes principales.

1) Tecnología altamente escalable: Un clúster Hadoop puede crecer añadiendo nuevos nodos, sin necesidad de modificar la estructura inicial, por lo que a diferencia de las bases de datos relacionadas es fácilmente escalable. Además, gracias al procesamiento distribuido de *MapReduce* los archivos se dividen en bloques de forma sencilla.

2) Velocidad: Hadoop permite ejecutar procesamientos y realizar análisis muy rápido, debido a su alta escalabilidad.

3) Flexibilidad: Al incrementar el número de nodos del sistema, aparte de ganar espacio, es posible agregar o acceder a nuevas y diferentes fuentes de datos. Al mismo tiempo existe la posibilidad de adaptar herramientas accesorias que funcionan en el entorno, ayudan en el diseño de procesos y la integración, entre otros posibles aspectos.

4) Almacenamiento a bajo costo: La información no se almacena de forma predefinida en filas y columnas como ocurre en las bases de datos tradicionales, sino que se le asignan datos categorizados a través de miles de computadoras, lo que supone un gran ahorro.

5) Tolerante a fallos: Debido a sus particiones o divisiones, si un equipo se cae y deja de funcionar por un tiempo, siempre existe una copia disponible, con lo que es posible la recuperación de datos en caso de producirse fallos.

2.2.3 Cloudera

Cloudera es una firma especializada en Big Data que permite añadir funciones de seguridad, control y gestión a la arquitectura HDFS. Su software está basado en el código abierto de Apache Hadoop, formando así la plataforma CDH (*Cloudera Distribution Hadoop*). No solo incluye el núcleo de Hadoop, sino que también integra diversos proyectos de Apache como, por ejemplo, Hive.

Figura 2.2.3: Cloudera (CDH)

2.3 Herramienta “rsync”

Rsync sirve para copiar archivos de forma rápida, ya sea localmente, hacia/desde otro host a través de cualquier *shell* remoto, o hacia/desde un *daemon* rsync remoto. Se trata de una herramienta extraordinariamente versátil, la cual ofrece una gran cantidad de opciones que controlan cada aspecto de su comportamiento, permitiendo una manipulación muy flexible del conjunto de archivos que se copiarán.

Figura 2.3: Ilustración del funcionamiento de rsync

Rsync destaca principalmente por su algoritmo de transferencia delta. Este algoritmo reduce la cantidad de datos enviados a través de la red, transfiriendo solo las diferencias entre los archivos de origen y los existentes en el destino. Se utiliza, sobretodo, para copias de seguridad y duplicación, así como un comando de copia mejorado para uso diario.

Su funcionamiento se basa en buscar archivos que han cambiado de tamaño o han sido modificados y, por lo tanto, tienen distinta hora de modificación. Cualquier cambio en el resto de atributos de un archivo origen se realiza directamente en el archivo destino cuando la comprobación indica que no es necesario actualizar los datos del archivo.

Seguidamente se nombran algunas de las características adicionales de Rsync.

- Proporciona soporte para copiar enlaces, propietarios, grupos y permisos.
- Puede usar cualquier *shell* remoto, incluidos ssh o rsh.
- No requiere permisos ni privilegios de super-usuario para poder ejecutarlo.
- Canaliza la transferencia de archivos para minimizar los costes de latencia.

Se utiliza a través de una Shell y su uso es,

“rsync <opciones> <ruta origen> <ruta destino>”

Las opciones básicas con las que cuenta son las siguientes:

- **“-v”:** muestra los resultados de la ejecución.
- **“-a”:** mantiene el usuario, grupo, permisos, enlaces simbólicos y la fecha y hora.
- **“-z”:** comprime la información antes de realizar la transferencia.
- **“-delete”:** borra la información que se encuentre en destino pero no en origen.
- **“-h”:** muestra las tasas de transferencia y el tamaño de los archivos.
- **“-progress”:** indica el porcentaje del archivo enviado y la tasa de transferencia.
- **“-stats”:** al final de la transferencia muestra un resumen con todos los datos más relevantes, como el número de archivos, archivos creados y eliminados, etc.

2.4 Herramienta “Resilio Sync”

Es una herramienta propietaria “peer-to-peer” de sincronización de archivos. Puede sincronizar datos entre dispositivos en una red local, o entre distintos terminales remotos a través de Internet, por medio de una versión modificada del protocolo BitTorrent. No es considerada como un reemplazo directo ni competidor de los servicios de sincronización basados en la nube, pero ha alcanzado gran parte de su publicidad en este papel potencial, debido a la capacidad que posee para abordar muchos de los problemas en los servicios existentes en relación con los límites de almacenamiento de archivos, la privacidad, el costo y el rendimiento.

Su funcionamiento se basa en almacenar los datos de usuario en un dispositivo local en lugar de la “nube”, por lo que requiere al menos dos dispositivos de usuario para estar en línea y sincronizar archivos entre ellos. Los datos se envían directamente entre los terminales conectados, a no ser que uno de ellos sea inalcanzable para el otro, en cuyo caso los datos se transmiten primero a un nodo intermediario, que deberá ser otro dispositivo conectado a ambos. Muchos terminales pueden estar conectados simultáneamente y compartir archivos entre ellos en una topología de red de malla.

2.5 Herramienta “Syncthing”

Se trata de un software multiplataforma escrito en el lenguaje de programación “Go”, que permite sincronizar y compartir archivos de manera sencilla entre todo tipo de dispositivos (ordenadores, tablets, móviles, etc.) de forma muy parecida a como lo hace Resilio Sync, es decir, de manera descentralizada, sin depender de servidores de terceros.

El programa ofrece tanto una interfaz de usuario web, para usar en cualquier navegador, como una “GUI” construida sobre “GTK” para escritorio. Además cuenta con un cliente para Android, lo que le permite ser utilizada en una gran mayoría de dispositivos móviles.

2.6 Herramienta “Nextcloud”

Programa con un completo software que permite sincronizar archivos, carpetas, calendarios y contactos entre múltiples dispositivos, permitiendo al usuario poseer el control total de todos sus datos, ya que dichos datos se almacenan en su red local, sin la necesidad de subir nada a una nube pública o servidor externo, a no ser que se quiera hacerlo.

Dicho software está centrado específicamente en proporcionar a todos sus usuarios seguridad, privacidad y control total, como se mencionaba anteriormente.

Se encuentra en continuo desarrollo por parte de sus desarrolladores y la comunidad que lo consume debido a que es de código abierto.

3 Definición del Proyecto

A lo largo de este capítulo se explica cuál es el alcance de RsyncHdfs y la metodología empleada durante su desarrollo.

3.1 Alcance

El público al que se dirige este Trabajo de Fin de Grado, son todos aquellos estudiantes o trabajadores que necesitan manejar numerosos archivos en HDFS para poder acumular gran cantidad de información, es decir, a todas aquellas personas que trabajan con Big Data a través de Hadoop utilizando un sistema operativo Linux.

RsyncHdfs es un script o ejecutable desarrollado en código Python cuyo funcionamiento sirve para sincronizar archivos locales de sistemas operativos Linux con archivos HDFS. Los usuarios podrán sincronizar todos los archivos que deseen de manera fácil e intuitiva, para poder manejarlos en el entorno Hadoop. De este modo, podrán trabajar con toda la información Big Data que utilicen de forma más rápida, ya que en Hadoop se pueden utilizar diferentes nodos de procesamiento.

Para saber qué ocurre durante la ejecución de RsyncHdfs, se genera un fichero "log" en el cual se graba la funcionalidad que se está llevando a cabo, así como todos los posibles errores que puedan surgir durante la sincronización si no finaliza de manera satisfactoria. De esta forma, el usuario podrá saber si la sincronización ha fallado por algún archivo en especial, ya sea porque el archivo está protegido, no posea los permisos necesarios, no exista, etc.

3.2 Metodología

Tras analizar las tecnologías existentes y el alcance que deberá tener el proyecto a desarrollar, se ha decidido que la metodología ágil más adecuada para RsyncHdfs es la que sigue un ciclo de vida incremental e iterativo.

El proyecto está compuesto por incrementos y para desarrollar cada uno de ellos se han llevado a cabo las siguientes actividades:

1. Análisis.
2. Diseño.
3. Codificación.
4. Pruebas Unitarias.
5. Pruebas de Integración.
6. Instauración y aprobación del uso del ejecutable.

Utilizando esta metodología, se obtienen resultados después de cada iteración, con los cuales se puede experimentar e ir ampliando y mejorando las funcionalidades paso a paso, obteniendo de esta manera una versión estable desde el primer incremento o primera versión del programa.

Las diferentes versiones se realizarán en el siguiente orden:

I. Versión 1.0: Función para el copiado de archivos desde Linux a HDFS.

II. Versión 1.1: Función para el copiado de archivos desde HDFS a Linux.

III. Versión 1.2: Función para la actualización de archivos existentes en Linux a partir de archivos procedentes de HDFS.

IV. Versión 1.3: Función para la actualización de archivos existentes en HDFS a partir de archivos procedentes de Linux.

V. Versión 1.4: Funciones para copiar y actualizar simultáneamente; para ambos sentidos de la sincronización.

VI. Versión 2.0: Funciones para borrar archivos en destino si no existen en origen.

VII. Versión 2.1: Funcionalidad añadida para eliminar archivos de origen si se han copiado u actualizado satisfactoriamente en destino.

VIII. Versión 3.0: Creación de la función que escribirá el log de la sincronización para tener una mayor información de la ejecución.

IX. Versión 3.1: Modularización de RsyncHdfs, creando un lanzador el cuál llamará al fichero/biblioteca en el cuál se encuentran las funciones definidas para cada utilidad.

4 Análisis de Requisitos

En este capítulo se presenta el análisis de los requisitos funcionales y no funcionales del proyecto. Dichos requisitos son los que definen las características finales del sincronizador RsyncHdfs. Debido a la metodología elegida, los requisitos se han ido modificando y ampliando a medida que se avanzaba en el desarrollo, hasta llegar al conjunto que se presenta en esta sección.

4.1 Requisitos funcionales

RF 1. Sincronizar directorios Linux con directorios HDFS de Hadoop.

RF 1.1. Directorios Linux a partir del contenido de directorios HDFS.

RF 1.2. Directorios HDFS a partir del contenido de directorios Linux.

RF 2. Comparar ficheros a sincronizar dentro del directorio.

RF 2.1. Comparar ficheros por nombre.

RF 2.1.1. Si un fichero no existe en el directorio que se está sincronizando, copiar dicho fichero cuándo se especifique la opción de copiado.

RF 2.1.2. Si un fichero ya existe en el directorio que se está sincronizando, se deberá reemplazar por el más actual siempre y cuándo se especifique la opción de actualizar y pueda ser reemplazado (RF 2.2.).

RF 2.2. Comparar ficheros por fecha de creación y última modificación.

RF 2.2.1. Si el fichero a sincronizar ya existe y se desea actualizar, se revisarán las fechas de creación y última modificación para saber cuál es el más actual y si se ha de reemplazar o no.

RF 3. Comparar carpetas a sincronizar dentro del directorio.

RF 3.1. Comparar carpetas por nombre.

RF 3.1.1. Si una carpeta no existe en uno de los directorios que se está sincronizando, pero en el otro si, copiar dicha carpeta cuándo se especifique la opción de copiado.

RF 3.1.2. Si una carpeta ya existe en ambos directorios, los ficheros que contenga la misma se compararán para saber si se han de actualizar (RF 2.), siempre que se especifique la opción de actualizar.

RF 4. Eliminar ficheros del directorio que se está sincronizando cuándo se especifique la opción de borrar, si no existen en el otro directorio.

RF 5. Actualizar ficheros del directorio que se está sincronizando cuándo se indique la opción de actualizar, si estos ya existen y se cumple RF 2.2.

RF 6. Crear o copiar nuevos ficheros en el directorio que se está sincronizando cuándo este activa la opción de copiar, si dichos ficheros no existen.

RF 7. Borrar ficheros del origen de la sincronización si se especifica la opción de borrar origen, una vez se halla sincronizado el directorio destino.

RF 8. Eliminar carpetas del directorio que se está sincronizando cuándo se especifique la opción de borrar, si no existen en el otro directorio.

RF 9. Actualizar carpetas del directorio que se está sincronizando cuándo se indique la opción de actualizar, si estas ya existen y se cumple RF 3.1.

RF 10. Crear o copiar nuevas carpetas en el directorio que se está sincronizando cuándo este activa la opción de copiar, si dichas carpetas no existen.

RF 11. Borrar carpetas del origen de la sincronización si se especifica la opción de borrar origen, una vez se halla sincronizado el directorio destino.

RF 12. Si no se especifican opciones (copiar, actualizar, borrar o borrar origen), el directorio que se está sincronizando deberá quedarse con exactamente el mismo contenido que el directorio origen.

RF 13. Si se especifican varias opciones a la vez, sincronizar el directorio cumpliendo todas las funcionalidades siempre que sea posible.

> **RF 13.1.** Copiar y actualizar: Se copian y se actualizan los ficheros y carpetas del directorio, cumpliendo con los requisitos RF 2.1.1., RF 2.2.1. Y RF 3.1.1.
>
> **RF 13.2.** Copiar y eliminar de destino: Se copian ficheros y carpetas nuevas y se eliminan los que existan en la carpeta destino y no estén en la carpeta origen, cumpliendo con los requisitos RF 2.1.2., RF 3.1.2., RF 4 y RF 8.
>
> **RF 13.3.** Copiar y eliminar de origen: Se copian ficheros y carpetas nuevas y se eliminan de la carpeta origen, cumpliendo con los requisitos RF 2.1.2., RF 3.1.2., RF 11.
>
> **RF 13.4.** Copiar, actualizar y eliminar de destino.
>
> **RF 13.5.** Copiar, actualizar y eliminar de origen.
>
> **RF 13.6.** Copiar, actualizar, eliminar de destino y origen.
>
> **RF 13.7.** Actualizar y eliminar.
>
> **RF 13.8.** Actualizar y eliminar de origen.
>
> **RF 13.9.** Actualizar, eliminar de destino y origen.
>
> **RF 13.10.** Eliminar de destino y origen.

RF 14. No se eliminarán ficheros o carpetas si no se especifica ninguna opción de borrado. El sincronizador no ha de eliminar ni modificar datos si el usuario no desea hacerlo.

RF 15. Los argumentos obligatorios que habrá que señalar en la ejecución son: -d, para indicar en qué sentido se llevará a cabo la sincronización, locToHdfs (para una ubicación origen en local, Linux y un destino en Hadoop) o hdfsToLoc (para partir de Hadoop a una ruta final local en Linux); -pL, para indicar la ubicación de origen, y, -pH, para señalar la ruta de destino.

RF 16. Las parámetros que podrá recibir de forma optativa RsyncHdfs son: -cp, para copiar archivos; -u, para actualizar archivos existentes, de acuerdo a las especificaciones indicadas; -rm, para eliminar archivos que existan en destino y no se encuentren en la ubicación de origen; -rmOri, si se quieren eliminar archivos del lugar de origen una vez sincronizados en destino, y, -log si se quiere especificar una ruta de almacenamiento concreta para el log que generará la ejecución.

4.1 Requisitos no funcionales

RNF 1. El sincronizador deberá ser fácil de utilizar (que no requiera de un gran esfuerzo de aprendizaje).

RNF 2. El sincronizador deberá ser efectivo, consiguiendo que los usuarios logren sus objetivos.

RNF 3. El sincronizador será consciente de las tareas que debe realizar dependiendo de las opciones que se especifiquen.

RNF 4. El tiempo de sincronización entre directorios dependerá del tamaño y la cantidad de archivos que deban sincronizarse.

> **RNF 4.1.** El tiempo de ejecución de la sincronización para un archivo individual no será superior a 20 minutos, si dicho archivo es de gran tamaño (varios GB).
>
> **RNF 4.2.** El tiempo de ejecución de la sincronización para un archivo individual no superará los 10 minutos, si dicho archivo no llega a 1 GB de capacidad.

RNF 5. Si existen archivos protegidos o sin los suficientes permisos para su manipulación, el sincronizador no realizará ninguna acción sobre ellos (Seguridad).

RNF 6. Si ocurre algún fallo durante la sincronización no se eliminará ningún archivo implicado en la misma.

> **RNF 6.1.** Si se ocasiona un error durante la ejecución de la sincronización, los ficheros quedarán como se encontraban antes de comenzar dicha sincronización.

RNF 7. Las opciones disponibles para la sincronización serán sencillas y comprensibles.

RNF 8. La sincronización quedará registrada en un log para saber qué es lo que ha ido ocurriendo durante su ejecución.

RNF 9. El sincronizador deberá ser eficiente en cuanto a las solicitudes de sus funcionalidades.

RNF 10. El sincronizador será un script Python, por lo tanto, para poder ejecutarlo será necesario tener Python instalado.

RNF 11. El sincronizador sólo funcionará en sistemas Linux con acceso a Hadoop.

RNF 12. Será necesario disponer de una versión estable HDFS compatible con el entorno Linux en el que se esté trabajando.

5 Diseño y Desarrollo

A continuación se describe el diseño del sincronizador, basado en el análisis realizado anteriormente. Además en este apartado se definirá con que tecnologías y software se ha desarrollado esta herramienta. Se verán los programas utilizados para su realización y el código elegido para la programación del sincronizador, como también otros programas usados para diversas funciones.

La herramienta creada se ha basado en un diseño realizado previamente que se fue refinando en distintas versiones de este proyecto, tal y como se explicó en el apartado **3 Definición del proyecto**, en las que se fue cambiando su modularización, hasta obtener el prototipo final. Primeramente, se realizó una identificación y verificación de los requisitos, con lo que se programó la primera versión (Versión 1.0), la cual se fue modificando hasta llegar a la versión actual (Versión 3.1).

5.1 Estructura y módulos

En primer lugar, se muestra la estructura y modularización interna del sincronizador RsyncHdfs, dividida en las partes que conforman cada uno de los dos archivos Python de que se compone. El lanzador de la sincronización (rsyncHdfsLauncher.py) contará con las distintas opciones para ejecutar el sincronizador, mientras que el sincronizador (Synchronicer.py) contendrá las distintas opciones para conseguir una ejecución satisfactoria. En el apartado 5.3 se explica cada una de las opciones disponibles.

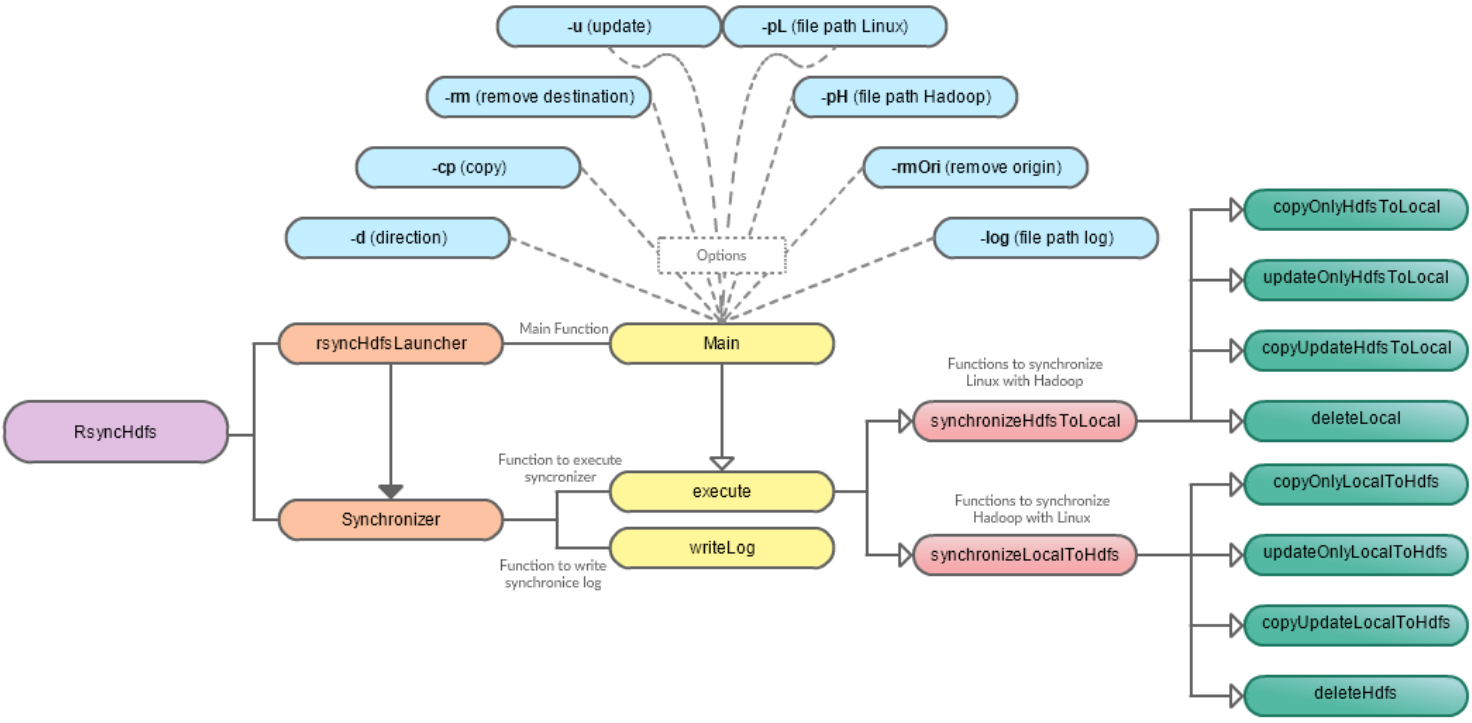

Figura 5.1: Estructura y modularización de RsyncHdfs

5.2 Tecnología y software utilizado

Para la elaboración de este proyecto, se ha de pensar antes de nada en los programas a utilizar para la edición de los ficheros de programación, y también en un programa o soporte para guardar y almacenar las versiones realizadas. Por otro lado, se utilizará una máquina virtual para alojar en ella el software necesario para el sincronizador, tanto el sistema operativo Linux como el Cloudera en el cual se encuentra Hadoop.

Seguidamente se lista el software manejado y se explica cuál ha sido exactamente su función, así como el lenguaje utilizado en la programación.

- *Python v2.7:* Lenguaje de programación utilizado para la creación y desarrollo del sincronizador. Se ha elegido este lenguaje de programación porque es uno de los más utilizados en la actualidad y cuenta con un gran soporte de actualizaciones.

- *VirtualBox v5.1:* Software de virtualización que permite crear máquinas virtuales. Ha sido utilizado para poder tener un entorno de pruebas con sistema operativo Linux y un Cloudera instalado en el que se alberga Hadoop.

- *CentOS v6.4:* Software con sistema operativo Linux para albergar los ficheros locales de la sincronización.

- *Cloudera v5.3.0:* Aloja el software de Hadoop.

- *Hadoop v2.5.0:* Sistema de almacenamiento distribuido de ficheros. Utilizado para almacenar grandes volúmnes de datos.

- *Eclipse IDE for Java Developers vJuno Service Release 2:* Entorno de desarrollo de código libre, utilizado para la creación de los ficheros Python, seleccionado por tener familiaridad con el entorno tras haber programado con él durante el Grado de Ingeniería Informática.

- *Dropbox v3.1.65.1:* Servicio de alojamiento de archivos en la nube. Se ha usado para guardar y almacenar las distintas versiones de la herramienta, para tenerlas accesibles en todo momento en cualquier lugar. Además se ha creado una carpeta compartida con la tutora para mantener una guía a lo largo del proyecto.

- *Notepad++ v7.4.1:* Editor de textos usado para la elaboración del documento "README" o manual de uso de la herramienta y para la edición de los ficheros Python desarrollados en Eclipse.

- *Microsoft Paint v1709:* Programa de Windows 10 utilizado para crear y modificar o recortar imágenes.

5.3 Implementación de las opciones para el usuario

Las distintas opciones con las que cuenta el usuario del sincronizador son copiado de archivos y/o actualización de los mismos; eliminación de archivos antiguos que ya no existen en uno de los extremos de la sincronización, o eliminación de ficheros que se han actualizado en destino y no queremos seguir conservando en origen.

Seguidamente se explican en detalle las funciones existentes para las opciones que posee el usuario, **mostrando la parte más relevante del código de las funciones** para cada opción de sincronización.

5.3.1 Copia de archivos de Linux a Hadoop

Como puede verse en el Código 5.2.1, dentro del recuadro verde se encuentra el código correspondiente a recorrer los archivos existentes en el directorio local de Linux especificado e insertarlos en una carpeta temporal que se copiará a Hadoop. La parte de código hallada dentro del recuadro azul, muestra la subida de los archivos a los directorios HDFS.

Se utiliza una carpeta temporal intermedia para que en caso de estar activa la opción de eliminar de origen, no se pierdan los datos si se produjera algún tipo de error en el copiado. Dicha carpeta se eliminará solo cuando la subida de archivos haya sido satisfactoria. De esta forma el sincronizador se asegura que no eliminará información si se produjera un fallo inesperado.

```
def copyOnlyLocalToHdfs(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
        deleteOrigin: Y delete files in origin if files be copied in destination
    """
    filesToCopy = False
    filesToMove = False
    listDirHdfs = []
    listDirAux = []
    folderSync = self.pathLocal.split('/')[len(self.pathLocal.split('/'))-1]

    try:
        self.writeLog('NOTE -- START ONLY COPY FROM LOCAL TO HDFS')
        dt = datetime.datetime.now()
        nameTempFolder = 'tempFolder_' + dt.strftime('%Y-%m-%d_%H.%M.%S.%f') + '_' + folderSync
        os.makedirs(self.tmpDir + '/' + str(nameTempFolder))
        for item in itemFolder:
            fileName = item["name"].split("/")
            listDirHdfs.append(fileName[-1])
        for file in listDir:
            if not file in listDirHdfs:
                if self.deleteOrigin == 'N':
                    filesToCopy = True
                    cmd = 'cp -u -p -R ' + self.pathLocal + '/' + file + ' '
                            + self.tmpDir + '/' + str(nameTempFolder)
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
                else:
                    try:
                        cmd = 'mv 2> /dev/null ' + self.pathLocal + '/' + file + ' '
                                + self.tmpDir + '/' + str(nameTempFolder)
                        sp.check_output(cmd, shell=True)
                        filesToMove = True
                    except:
                        filesToCopy = True
                        cmd = 'cp -u -p -R ' + self.pathLocal + '/' + file + ' '
                                + self.tmpDir + '/' + str(nameTempFolder)
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                        listDirAux.append(file)
        try:
            cmd = 'hadoop fs -copyFromLocal -p 2> ' + self.tmpDir + '/warnings.txt file://'
                    + self.tmpDir + '/' + str(nameTempFolder) + ' hdfs://' + self.masterNode + ':' + self.pathHdfs
            p = sp.Popen(cmd, shell=True)
            p.wait()
            itemTempFolder = conHdfs.list_directory(self.pathHdfs + '/' + str(nameTempFolder))
            if itemTempFolder != []:
                if filesToMove and os.path.isfile(self.tmpDir + '/warnings.txt'):
                    cmd = 'hadoop fs -chmod 774 hdfs://' + self.masterNode + ':' + self.pathHdfs + '/'
                            + str(nameTempFolder) + '/*'
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
```

Código 5.3.1: Función de copiado de archivos de Linux a Hadoop

5.3.2 Borrado de archivos de Hadoop

Se utiliza esta función cuando se realiza una sincronización con la opción de eliminar archivos que no existen en local (rm), sincronizando archivos de Linux con archivos Hadoop. Si existen ficheros en HDFS que no se encuentran en Linux, estos se eliminan para dejar ambos *paths* sincronizados, con el mismo número de archivos. Se recorre el directorio local buscando los ficheros que halla en Hadoop. Los que no existan se eliminan de HDFS.

```
def deleteHdfs(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
    """
    try:
        self.writeLog('NOTE -- START DELETE HDFS IF FILE OR DIRECTORY NOT EXIST IN LOCAL')
        for item in itemFolder:
            if item["kind"] == 'file':
                fileName = item["name"].split("/")
                if not fileName[-1] in listDir:
                    cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://'
                            + self.masterNode + ':' + self.pathHdfs + '/' + fileName[-1]
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
            elif item["kind"] == 'directory':
                dirName = item["name"].split("/")
                if not dirName[-1] in listDir:
                    cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://'
                            + self.masterNode + ':' + self.pathHdfs + '/' + dirName[-1]
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
                else:
                    listSubDir = os.listdir(self.pathLocal + '/' + dirName[-1])
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + dirName[-1])
                    for itemSub in itemSubFolder:
                        if itemSub["kind"] == 'file':
                            fileSubName = itemSub["name"].split("/")
                            if not fileSubName[-1] in listSubDir:
                                cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://'
                                        + self.masterNode + ':' + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
                                p = sp.Popen(cmd, shell=True)
                                p.wait()
```

Código 5.3.2: Función de borrado de archivos en Hadoop

5.3.3 Actualización de archivos en Hadoop a partir de archivos de Linux

Para actualizar ficheros existentes en Hadoop a partir de archivos más actuales que se encuentren en Linux, se realiza una comprobación de tiempo para saber si realmente los datos de origen son más recientes que los hallados en destino. Dicho tiempo se obtiene mediante el comando "stat", tal y como puede verse en el recuadro verde del Código 5.2.3. Tras la obtención de las fechas se realiza la comprobación para copiar o no el fichero (recuadro azul).

Por último, la parte de código enmarcada en amarillo muestra el mismo procedimiento para comprobar la fecha de subdirectorios.

```
def updateOnlyLocalToHdfs(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
    """
    try:
        self.writeLog('NOTE -- START ONLY UPDATE FROM LOCAL TO HDFS')
        listDirHdfs = []
        for item in itemFolder:
            fileName = item["name"].split("/")
            listDirHdfs.append(fileName[-1])
        for file in listDir:
            if file in listDirHdfs:
                dir = os.path.isdir(self.pathLocal + '/' + file)
                if dir == False:
                    cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + file
                    timeLocal = sp.check_output(cmd, shell=True)
                    cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + file
                    timeHadoop = sp.check_output(cmd, shell=True)
                    if timeHadoop < timeLocal:
                        cmd = 'hadoop fs -copyFromLocal -p -f file:// ' + self.pathLocal + '/' + file
                                + ' hdfs://' + self.masterNode + ':' + self.pathHdfs
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                else:
                    listSubDir = os.listdir(self.pathLocal + '/' + file)
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + file)
                    listSubDirHdfs = []
                    for itemSub in itemSubFolder:
                        fileSubName = itemSub["name"].split("/")
                        listSubDirHdfs.append(fileSubName[-1])
                    for fileSub in listSubDir:
                        if fileSub in listSubDirHdfs:
                            cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + file + '/' + fileSub
                            timeLocal = sp.check_output(cmd, shell=True)
                            cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + file + '/' + fileSub
                            timeHadoop = sp.check_output(cmd, shell=True)
                            if timeHadoop < timeLocal:
                                cmd = 'hadoop fs -copyFromLocal -p -f file:// ' + self.pathLocal + '/' + file
                                        + '/' + fileSub + ' hdfs://' + self.masterNode + ':' + self.pathHdfs + '/' + file
                                p = sp.Popen(cmd, shell=True)
                                p.wait()
```

Código 5.3.3: Función de actualización de archivos en Hadoop

5.3.4 Copia y actualización de archivos en Hadoop a partir de archivos de Linux

Se puede observar que cuando se desea copiar archivos desde Linux a HDFS, inexistentes en Hadoop, y al mismo tiempo actualizar los que existan comunes en ambos paths, tanto origen como destino, lo que se realiza es una combinación de las funciones ilustradas en las imágenes "Código 5.2.1" y "Código 5.2.3". Se comprueban las fechas de los ficheros comunes para saber si se deben o no actualizar y se realiza la copia directa de los no comunes desde el directorio Linux al directorio Hadoop.

```
def copyUpdateLocalToHdfs(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
    """
    try:
        self.writeLog('NOTE -- START COPY AND UPDATE FROM LOCAL TO HDFS')
        listDirHdfs = []
        for item in itemFolder:
            fileName = item["name"].split("/")
            listDirHdfs.append(fileName[-1])
        for file in listDir:
            if file in listDirHdfs:
                dir = os.path.isdir(self.pathLocal + '/' + file)
                if dir == False:
                    cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + file
                    timeLocal = sp.check_output(cmd, shell=True)
                    cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + file
                    timeHadoop = sp.check_output(cmd, shell=True)
                    if timeHadoop < timeLocal:
                        cmd = 'hadoop fs -copyFromLocal -p -f file://' + self.pathLocal + '/' + file
                                + ' hdfs://' + self.masterNode + ':' + self.pathHdfs
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                else:
                    listSubDir = os.listdir(self.pathLocal + '/' + file)
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + file)
                    listSubDirHdfs = []
                    for itemSub in itemSubFolder:
                        fileSubName = itemSub["name"].split("/")
                        listSubDirHdfs.append(fileSubName[-1])
                    for fileSub in listSubDir:
                        if fileSub in listSubDirHdfs:
                            cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + file + '/' + fileSub
                            timeLocal = sp.check_output(cmd, shell=True)
                            cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + file + '/' + fileSub
                            timeHadoop = sp.check_output(cmd, shell=True)
                            if timeHadoop < timeLocal:
                                cmd = 'hadoop fs -copyFromLocal -p -f file://' + self.pathLocal + '/' + file + '/' + fileSub
                                        + ' hdfs://' + self.masterNode + ':' + self.pathHdfs + '/' + file
                                p = sp.Popen(cmd, shell=True)
                                p.wait()
                        else:
                            cmd = 'hadoop fs -copyFromLocal -p file://' + self.pathLocal + '/' + file + '/' + fileSub
                                    + ' hdfs://' + self.masterNode + ':' + self.pathHdfs + '/' + file
                            p = sp.Popen(cmd, shell=True)
                            p.wait()
            else:
                cmd = 'hadoop fs -copyFromLocal -p file://' + self.pathLocal + '/' + file
                        + ' hdfs://' + self.masterNode + ':' + self.pathHdfs
                p = sp.Popen(cmd, shell=True)
                p.wait()
```

Código 5.3.4: Función de copiado y actualización de archivos en Hadoop a partir de ficheros en Linux

5.3.5 Copia de archivos de Hadoop a Linux

Se recorren los archivos del directorio en Hadoop para comprobar si existen en Linux. En la primera parte de código de esta función puede verse como se realiza el copiado de ficheros, que no existen en local, desde HDFS.

La segunda parte es similar, pero para el caso de directorios contenidos dentro del directorio al cual se le está realizando la sincronización.

```
def copyOnlyHdfsToLocal(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
        deleteOrigin: Y delete files in origin if files be copied in destination
    """
    try:
        self.writeLog('NOTE -- START ONLY COPY FROM HDFS TO LOCAL')
        for item in itemFolder:
            if item["kind"] == 'file':
                fileName = item["name"].split("/")
                if not fileName[-1] in listDir:
                    cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                            + self.pathHdfs + '/' + fileName[-1] + ' file://' + self.pathLocal
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
                    newListDir = os.listdir(self.pathLocal)
                    if self.deleteOrigin == 'Y' and fileName[-1] in newListDir:
                        cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://' + self.masterNode + ':'
                                + self.pathHdfs + '/' + fileName[-1]
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
            elif item["kind"] == 'directory':
                dirName = item["name"].split("/")
                if dirName[-1] in listDir:
                    listSubDir = os.listdir(self.pathLocal + '/' + dirName[-1])
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + dirName[-1])
                    for itemSub in itemSubFolder:
                        if itemSub["kind"] == 'file':
                            fileSubName = itemSub["name"].split("/")
                            if not fileSubName[-1] in listSubDir:
                                cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                                        + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
                                        + ' file://' + self.pathLocal + '/' + dirName[-1]
                                p = sp.Popen(cmd, shell=True)
                                p.wait()
                                newListSubDir = os.listdir(self.pathLocal + '/' + dirName[-1])
                                if self.deleteOrigin == 'Y' and fileSubName[-1] in newListSubDir:
                                    cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://' + self.masterNode + ':'
                                            + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
                                    p = sp.Popen(cmd, shell=True)
                                    p.wait()
                else:
                    cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                            + self.pathHdfs + '/' + dirName[-1] + ' file://' + self.pathLocal
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
                    newListDir = os.listdir(self.pathLocal)
                    if self.deleteOrigin == 'Y' and dirName[-1] in newListDir:
                        cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://' + self.masterNode + ':'
                                + self.pathHdfs + '/' + dirName[-1]
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
```

Código 5.3.5: Función de copiado de archivos de Hadoop a Linux

5.3.6 Borrado de archivos de Linux

Al ejecutar el sincronizador con la opción para borrar archivos existentes en destino, pero no en origen, en sentido de Hadoop a Linux, se busca cada uno de los ficheros locales en una lista que contiene los archivos existentes en HDFS.

Si dichos ficheros locales se encuentran en la lista nombrada, se mantienen y no se excluyen, en caso contrario, cuando no se encuentran, se eliminan para dejar ambos *paths* sincronizados, con el mismo número de archivos.

```
def deleteLocal(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
    """
    try:
        self.writeLog('NOTE -- START DELETE LOCAL IF FILE OR DIRECTORY NOT EXIST IN HDFS')
        listDirHdfs = []
        for item in itemFolder:
            fileName = item["name"].split("/")
            listDirHdfs.append(fileName[-1])
        for file in listDir:
            if not file in listDirHdfs:
                cmd = 'rm -r -f ' + self.pathLocal + '/' + file
                p = sp.Popen(cmd, shell=True)
                p.wait()
            else:
                dir = os.path.isdir(self.pathLocal + '/' + file)
                if dir == True:
                    listSubDir = os.listdir(self.pathLocal + '/' + file)
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + file)
                    listSubDirHdfs = []
                    for itemSub in itemSubFolder:
                        fileSubName = itemSub["name"].split("/")
                        listSubDirHdfs.append(fileSubName[-1])
                    for fileSub in listSubDir:
                        if not fileSub in listSubDirHdfs:
                            cmd = 'rm -r -f ' + self.pathLocal + '/' + file + '/' + fileSub
                            p = sp.Popen(cmd, shell=True)
                            p.wait()
```

Código 5.3.6: Función de borrado de archivos en Linux

5.3.7 Actualización de archivos en Linux a partir de archivos de Hadoop

Si se desea solo actualizar ficheros existentes en Linux a partir de archivos más actuales que se encuentren en Hadoop, se comprueba la fecha y tiempo de los ficheros para saber si realmente los datos de origen son más recientes que los hallados en destino. Como ya se mencionó anteriormente, Dicho tiempo se obtiene mediante el comando "stat". Una vez obtenidas las fechas se realiza la comprobación para copiar o no el fichero (recuadro azul), o el directorio (recuadro amarillo).

```
def updateOnlyHdfsToLocal(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
    """
    try:
        self.writeLog('NOTE -- START ONLY UPDATE FROM HDFS TO LOCAL')
        for item in itemFolder:
            if item["kind"] == 'file':
                fileName = item["name"].split("/")
                if fileName[-1] in listDir:
                    cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + fileName[-1]
                    timeLocal = sp.check_output(cmd, shell=True)
                    cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + fileName[-1]
                    timeHadoop = sp.check_output(cmd, shell=True)
                    if timeLocal < timeHadoop:
                        cmd = 'rm -r -f ' + self.pathLocal + '/' + fileName[-1]
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                        cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                                + self.pathHdfs + '/' + fileName[-1] + ' file://' + self.pathLocal
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                        newListDir = os.listdir(self.pathLocal)
                        if self.deleteOrigin == 'Y' and fileName[-1] in newListDir:
                            cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://' + self.masterNode + ':'
                                    + self.pathHdfs + '/' + fileName[-1]
                            p = sp.Popen(cmd, shell=True)
                            p.wait()
            elif item["kind"] == 'directory':
                dirName = item["name"].split("/")
                if dirName[-1] in listDir:
                    listSubDir = os.listdir(self.pathLocal + '/' + dirName[-1])
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + dirName[-1])
                    for itemSub in itemSubFolder:
                        if itemSub["kind"] == 'file':
                            fileSubName = itemSub["name"].split("/")
                            if fileSubName[-1] in listSubDir:
                                cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + dirName[-1] + '/' + fileSubName[-1]
                                timeLocal = sp.check_output(cmd, shell=True)
                                cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
                                timeHadoop = sp.check_output(cmd, shell=True)
                                if timeLocal < timeHadoop:
                                    cmd = 'rm -r -f ' + self.pathLocal + '/' + dirName[-1] + '/' + fileSubName[-1]
                                    p = sp.Popen(cmd, shell=True)
                                    p.wait()
                                    cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                                            + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
                                            + ' file://' + self.pathLocal + '/' + dirName[-1]
                                    p = sp.Popen(cmd, shell=True)
                                    p.wait()
```

Código 5.3.7: Función de actualización de archivos en Linux

5.3.8 Copia y actualización de archivos en Linux a partir de archivos de Hadoop

Como se ve en Código 5.3.8, si se quiere copiar archivos nuevos y a la vez actualizar los existentes en Linux, desde Hadoop, se utiliza una combinación de las funciones de copiado y actualizado de archivos.

```
def copyUpdateHdfsToLocal(self, conHdfs, itemFolder, listDir):
    """
    Args:
        conHdfs: hdfs functions
        itemFolder: list hdfs files
        listDir: list local files
        pathLocal:  path with local directory
        pathHdfs: path with hdfs directory
    """
    try:
        self.writeLog('NOTE -- START COPY AND UPDATE FROM HDFS TO LOCAL')
        for item in itemFolder:
            if item["kind"] == 'file':
                fileName = item["name"].split("/")
                if fileName[-1] in listDir:
                    cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + fileName[-1]
                    timeLocal = sp.check_output(cmd, shell=True)
                    cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + fileName[-1]
                    timeHadoop = sp.check_output(cmd, shell=True)
                    if timeLocal < timeHadoop:
                        cmd = 'rm -r -f ' + self.pathLocal + '/' + fileName[-1]
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                        cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                                + self.pathHdfs + '/' + fileName[-1] + ' file://' + self.pathLocal
                        p = sp.Popen(cmd, shell=True)
                        p.wait()
                else:
                    cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                            + self.pathHdfs + '/' + fileName[-1] + ' file://' + self.pathLocal
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
                newListDir = os.listdir(self.pathLocal)
                if self.deleteOrigin == 'Y' and fileName[-1] in newListDir:
                    cmd = 'hadoop fs -rm -r -f -skipTrash hdfs://' + self.masterNode + ':'
                            + self.pathHdfs + '/' + fileName[-1]
                    p = sp.Popen(cmd, shell=True)
                    p.wait()
            elif item["kind"] == 'directory':
                dirName = item["name"].split("/")
                if dirName[-1] in listDir:
                    listSubDir = os.listdir(self.pathLocal + '/' + dirName[-1])
                    itemSubFolder = conHdfs.list_directory(self.pathHdfs + '/' + dirName[-1])
                    for itemSub in itemSubFolder:
                        if itemSub["kind"] == 'file':
                            fileSubName = itemSub["name"].split("/")
                            if fileSubName[-1] in listSubDir:
                                cmd = 'stat -c "%Z" ' + self.pathLocal + '/' + dirName[-1] + '/' + fileSubName[-1]
                                timeLocal = sp.check_output(cmd, shell=True)
                                cmd = 'hadoop fs -stat "%Y" ' + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
                                timeHadoop = sp.check_output(cmd, shell=True)
                                if timeLocal < timeHadoop:
                                    cmd = 'rm -r -f ' + self.pathLocal + '/' + dirName[-1] + '/' + fileSubName[-1]
                                    p = sp.Popen(cmd, shell=True)
                                    p.wait()
                                    cmd = 'hadoop fs -copyToLocal -p -ignoreCrc hdfs://' + self.masterNode + ':'
                                            + self.pathHdfs + '/' + dirName[-1] + '/' + fileSubName[-1]
```

Código 5.3.8: Función de copiado y actualización de archivos en Linux a partir de ficheros en Hadoop

6 Pruebas y Resultados

En este capítulo se describen todas las pruebas que se han ido realizando a lo largo del desarrollo y los resultados finales obtenidos en el proyecto.

6.1 Pruebas de la herramienta RsyncHdfs

A lo largo del proceso de realización de RsyncHdfs se han establecido una serie de pruebas para verificar el correcto funcionamiento de la herramienta. Se ha prestado especial atención a las pruebas de verificación y validación para ofrecer al usuario un software funcional y útil.

Las pruebas de validación tienen como objetivo validar la funcionalidad requerida, es decir, comprobar que se cumplen todos los requisitos descritos en la Sección 4. Por otro lado, las pruebas de verificación certifican que los resultados obtenidos son los esperados.

6.1.1 Pruebas lógicas

El proceso de verificación consiste en asegurarse que el funcionamiento del software elaborado es el esperado. Para poder validar este proceso se han llevado a cabo inspecciones de código, pruebas de caja blanca y pruebas de caja negra.

- **Inspecciones de código:**

  En dichas inspecciones se revisa el código implementado. Las revisiones se realizaban en el final de cada iteración, detectando así posibles errores, con la posibilidad de solucionarlos antes del comienzo de la siguiente iteración.

  En el transcurso de estas pruebas se consideró oportuno repetir las inspecciones cada vez que se añadían nuevas funcionalidades o se modificaban funcionalidades ya existentes, revisando de esta forma el código generado en las etapas anteriores.

- **Pruebas de caja blanca:**

  Las pruebas de caja blanca, también conocidas como pruebas estructurales o de cobertura lógica se basan en revisar el correcto funcionamiento interno de la herramienta.

  El diseño de estas pruebas se basa en el camino que ha de seguir el código y el estudio de las variables del programa.

  Se creó una batería de pruebas para cada una de las funcionalidades del sincronizador. Tras la ejecución de la misma se corrigieron los errores encontrados, relacionados sobre todo con el control de funcionalidades que se iban añadiendo.

- **Pruebas de caja negra:**

Las pruebas de caja negra o pruebas de comportamiento consisten en la comprobación del buen funcionamiento de la herramienta, sin importar los pasos que se han realizado para llevarla a cabo.

Se ha creado una batería de pruebas cubriendo todos los casos posibles, introduciendo datos normales y extremos en la ejecución. Para estas pruebas se aplica un determinado grupo de entradas y se observa si las salidas obtenidas para cada función son las esperadas, comprobando así que todo funciona correctamente.

6.1.2 Pruebas de interacción con la herramienta

Se han explorado los directorios origen y destino antes de la sincronización y una vez realizada, para corroborar que tras la ejecución los archivos finales en ambas rutas son exactamente los esperados, aunque serán los usuarios finales quienes tengan el último voto de aprobación sobre su grado de cumplimiento, tal y como ocurre con los requisitos de uso.

6.2 Pruebas de funcionamiento y resultados

Tras la implementación y desarrollo de la herramienta se han ejecutado diferentes pruebas con el fin de validar su funcionalidad y usabilidad. Las pruebas están enfocadas para los usuarios a los que está destinado el sincronizador RsyncHdfs: estudiantes o trabajadores que necesitan manejar gran cantidad de datos mediante Big Data en Hadoop, usando un sistema operativo Linux.

A continuación se muestra el contenido de los directorios origen y destino utilizados para la realización de dichas pruebas, tras realizar la sincronización con las opciones indicadas.

6.2.1 Copiado de archivos existentes en Linux a Hadoop

La Tabla 6.2.1 muestra los archivos contenidos tanto en Linux como en Hadoop antes y después de ejecutar el sincronizador con la opción -cp: "*python rsyncHdfs.py -d loctohdfs -pL <path en Linux de Apuntes> -pH <path en Hadoop de Apuntes> -cp Y*".

Nombre del Directorio	***Apuntes***	
Sincronización	**Antes**	**Después**
Ficheros en Linux	*Circuitos.odt* *Estructura.odt* *Fundamentos.odt*	*Circuitos.odt* *Estructura.odt* *Fundamentos.odt*
Ficheros en Hadoop	*Estructuras.odt* *Fundamentos.odt*	*Circuitos.odt* *Estructura.odt* *Fundamentos.odt*

Tabla 6.2.1: Copia de archivos existentes en Linux a Hadoop

6.2.2 Actualización de archivos de Hadoop a partir de ficheros existentes en Linux

En la Tabla 6.2.2 se pueden observar los resultados tras la ejecución de: "*python rsyncHdfs.py -d loctohdfs -pL <path en Linux de Notas> -pH <path en Hadoop de Notas> -u Y*".

Nombre del Directorio	***Notas***	
Sincronización	**Antes**	**Después**
Ficheros en Linux	*NotasPracticas.xlsx - (11/04/2018 09:44)* *NotasTeoria.xlsx - (11/04/2018 09:43)*	*NotasPracticas.xlsx - (11/04/2018 09:44)* *NotasTeoria.xlsx - (11/04/2018 09:43)*
Ficheros en Hadoop	*NotasPracticas.xlsx - (11/04/2018 08:53)* *NotasTeoria.xlsx - (11/04/2018 08:53)*	*NotasPracticas.xlsx - (11/04/2018 09:44)* *NotasTeoria.xlsx - (11/04/2018 09:43)*

Tabla 6.2.2: Actualización de archivos de Hadoop

6.2.3 Copiado de archivos existentes en Linux a Hadoop y actualización de ficheros de Hadoop a partir de los hallados en Linux

La Tabla 6.2.3 recoge los resultados de las 2 opciones anteriores combinadas.

Nombre del Directorio	***Examenes***	
Sincronización	**Antes**	**Después**
Subdirectorios en Linux	*Practicas* *Teoria*	*Practicas* *Teoria*
Subdirectorios en Hadoop	*Practicas*	*Practicas* *Teoria*
Ficheros en Linux (subdirectorios)	*PSI.zip - (11/04/2018 09:30)* *Pautlen.zip - (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)* *Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*	*PSI.zip - (11/04/2018 09:30)* *Pautlen.zip - (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)* *Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*
Ficheros en Hadoop (subdirectorios)	*PSI.zip - (11/04/2018 08:41)* *Pautlen.zip – (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)*	*PSI.zip - (11/04/2018 09:30)* *Pautlen.zip - (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)* *Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*

Tabla 6.2.3: Copia y actualización de archivos en Hadoop

6.2.4 Copiado de archivos existentes en Linux a Hadoop y eliminación de los existentes en Hadoop no hallados en Linux

En la Tabla 6.2.4 se pueden observar los ficheros contenidos en local (Linux) como en HDFS (Hadoop) antes y después de ejecutar el sincronizador con las opciones *copy* y *remove*: "*python rsyncHdfs.py -d loctohdfs -pL <path en Linux de Practicas> -pH <path en Hadoop de Practicas> -cp Y -rm Y*".

Nombre del Directorio	***Practicas***	
Sincronización	**Antes**	**Después**
Ficheros en Linux	*Moviles.zip* *Prog2.tar.gz* *RedesI.tar* *Videojuegos.zip*	*Moviles.zip* *Prog2.tar.gz* *RedesI.tar* *Videojuegos.zip*
Ficheros en Hadoop	*Moviles_Viejo.gz* *Prog2.tar.gz* *PruebasMoviles.tar* *RedesI.tar* *Videojuegos.zip*	*Moviles.zip* *Prog2.tar.gz* *RedesI.tar* *Videojuegos.zip*

Tabla 6.2.4: Copia y eliminación de archivos en Hadoop

6.2.5 Copiado de archivos existentes en Linux a Hadoop, eliminación de los existentes en Hadoop no hallados en Linux y eliminación en Linux de los copiados

La Tabla 6.2.5 muestra los archivos contenidos en Linux y Hadoop antes y después de ejecutar el sincronizador con la opción -cp: "*python rsyncHdfs.py -d loctohdfs -pL <path en Linux de Trabajo> -pH <path en Hadoop de Trabajo> -cp Y -rm Y-–rmOri Y*".

Nombre del Directorio	***Trabajos***	
Sincronización	**Antes**	**Después**
Subdirectorios en Linux	*Obligatorios* *Optativos*	*Obligatorios*
Subdirectorios en Hadoop	*Obligatorios*	*Obligatorios* *Optativos*
Ficheros en Linux (subdirectorios)	*Algebra.odt* *Electromagnetismo.odt* *PINGS.odt* *InteligenciaArtificial.odt* *ProbabilidadyEstadisticaEmpresarial.odt*	*Algebra.odt* *Electromagnetismo.odt* *PINGS.odt*
Ficheros en Hadoop (subdirectorios)	*Algebra.odt* *Algebra_copia.odt* *Calculo.odt* *Electromagnetismo.odt* *PINGS.odt*	*Algebra.odt* *Electromagnetismo.odt* *PINGS.odt* *InteligenciaArtificial.odt* *ProbabilidadyEstadisticaEmpresarial.od*

Tabla 6.2.5: Copia y eliminación de archivos en Hadoop y eliminación de archivos en Linux tras la copia

6.2.6 Copiado de archivos existentes en Linux a Hadoop, actualización de ficheros de Hadoop a partir de los hallados en Linux y eliminación en Linux de los copiados o actualizados

El comando ejecutado para los resultados de la Tabla 6.2.6 es el siguiente: "*python rsyncHdfs.py -d loctohdfs -pL <path en Linux de Universidad> -pH <path en Hadoop de Universidad> -cp Y -u Y-rmOri Y*".

Nombre del Directorio	***Universidad***	
Sincronización	**Antes**	**Después**
Subdirectorios en Linux	*Apuntes - (11/04/2018 09:25)* *Examenes - (11/04/2018 09:37)* *Notas - (11/04/2018 10:16)* *Practicas - (11/04/2018 10:17)* *Trabajos - (11/04/2018 10:18)*	-
Subdirectorios en Hadoop	*Apuntes - (11/04/2018 08:35)* *Examenes - (11/04/2018 09:11)* *Notas - (11/04/2018 09:44)* *Practicas - (11/04/2018 09:54)* *Trabajos - (11/04/2018 10:08)*	*Apuntes - (11/04/2018 09:25)* *Examenes - (11/04/2018 09:37)* *Notas - (11/04/2018 10:16)* *Practicas - (11/04/2018 10:17)* *Trabajos - (11/04/2018 10:18)*
Ficheros en Linux	*Cosas_Varias_Uni.zip*	-
Ficheros en Hadoop	-	*Cosas_Varias_Uni.zip*

Tabla 6.2.6: Copia y actualización de archivos en Hadoop y eliminación de archivos en Linux tras la copia o actualización

6.2.7 Copiado de archivos existentes en Hadoop a Linux

Los resultados de la Tabla 6.2.7 se obtienen tras la ejecución de la misma sentencia que se especificaba en la Tabla 6.2.1, cambiando la dirección por: "*-d hdfstoloc*".

Nombre del Directorio	***Apuntes***	
Sincronización	**Antes**	**Después**
Ficheros en Hadoop	*Circuitos.odt* *Estructura.odt* *Fundamentos.odt*	*Circuitos.odt* *Estructura.odt* *Fundamentos.odt*
Ficheros en Linux	*Circuitos.odt*	*Circuitos.odt* *Estructura.odt* *Fundamentos.odt*

Tabla 6.2.7: Copia de archivos existentes en Hadoop a Linux

6.2.8 NO actualización de archivos de Linux a partir de ficheros existentes en Hadoop

Ejecución del comando/sentencia: "*python rsyncHdfs.py -d hdfstoloc -pL <path en Linux de Notas> -pH <path en Hadoop de Notas> -u Y".*

NOTA: Los ficheros de Linux no se actualizan debido a que son más modernos.

Nombre del Directorio	***Notas***	
Sincronización	**Antes**	**Después**
Ficheros en Hadoop	*NotasPracticas.xlsx - (11/04/2018 09:44)* *NotasTeoria.xlsx - (11/04/2018 09:43)*	*NotasPracticas.xlsx - (11/04/2018 09:44)* *NotasTeoria.xlsx - (11/04/2018 09:43)*
Ficheros en Linux	*NotasPracticas.xlsx - (16/04/2018 09:17)* *NotasTeoria.xlsx - (16/04/2018 09:18)*	*NotasPracticas.xlsx - (16/04/2018 09:17)* *NotasTeoria.xlsx - (16/04/2018 09:18)*

Tabla 6.2.8: NO actualización de archivos de Linux

6.2.9 Copiado de archivos existentes en Linux a Hadoop y actualización de ficheros de Hadoop a partir de los hallados en Linux

La Tabla 6.2.9 recoge los resultados de las 2 opciones anteriores combinadas.

Nombre del Directorio	***Examenes***	
Sincronización	**Antes**	**Después**
Subdirectorios en Hadoop	*Practicas* *Teoria*	*Practicas* *Teoria*
Subdirectorios en Linux	*Practicas*	*Practicas* *Teoria*
Ficheros en Hadoop (subdirectorios)	*PSI.zip - (11/04/2018 09:30)* *Pautlen.zip - (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)* *Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*	*PSI.zip - (11/04/2018 09:30)* *Pautlen.zip - (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)* *Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*
Ficheros en Linux (subdirectorios)	*Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*	*PSI.zip - (11/04/2018 09:30)* *Pautlen.zip - (11/04/2018 08:40)* *Soper.zip - (11/04/2018 08:40)* *Redes1.zip - (11/04/2018 08:37)* *Redes2.zip - (11/04/2018 08:38)*

Tabla 6.2.9: Copia y actualización de archivos en Hadoop

6.2.10 Copiado de archivos existentes en Hadoop a Linux y eliminación de los existentes en Linux no hallados en Hadoop

La Tabla 6.2.10 muestra los archivos contenidos tanto en Linux como en Hadoop antes y después de ejecutar el sincronizador con la opción -cp: "*python rsyncHdfs.py -d hdfstoloc -pL <path en Linux de Practicas> -pH <path en Hadoop de Practicas> -cp Y -rm Y".*

Nombre del Directorio	***Practicas***	
Sincronización	**Antes**	**Después**
Ficheros en Hadoop	*Moviles.zip* *Prog2.tar.gz* *RedesI.tar* *Videojuegos.zip*	*Moviles.zip* *Prog2.tar.gz* *RedesI.tar* *Videojuegos.zip*
Ficheros en Linux	*Moviles.zip* *Prog2.tar.gz* *RedesI_antiguo.tar* *RedesI.tar* *Videojuegos_antiguo.zip*	*Moviles.zip* *Prog2.tar.gz* *RedesI.tar* *Videojuegos.zip*

Tabla 6.2.10: Copia y eliminación de archivos en Linux

6.2.11 Copiado de archivos existentes en Hadoop a Linux, eliminación de los existentes en Linux no hallados en Hadoop y eliminación en Hadoop de los copiados

En la Tabla 6.2.11 se pueden observar los resultados tras la ejecución de: "*python rsyncHdfs.py -d hdfstoloc -pL <path en Linux de Trabajos> -pH <path en Hadoop de Trabajos> -cp Y -rm Y -rmOri Y".*

Nombre del Directorio	***Trabajos***	
Sincronización	**Antes**	**Después**
Subdirectorios en Hadoop	*Obligatorios* *Optativos*	*Optativos*
Subdirectorios en Linux	*Optativos*	*Obligatorios* *Optativos*
Ficheros en Hadoop (subdirectorios)	*Algebra.odt* *Electromagnetismo.odt* *PINGS.odt* *InteligenciaArtificial.odt* *ProbabilidadyEstadisticaEmpresarial.odt*	*InteligenciaArtificial.odt*
Ficheros en Linux (subdirectorios)	*InteligenciaArtificial.odt*	*Algebra.odt* *Electromagnetismo.odt* *PINGS.odt* *InteligenciaArtificial.odt* *ProbabilidadyEstadisticaEmpresarial.odt*

Tabla 6.2.11: Copia y eliminación de archivos en Linux y eliminación de archivos en Hadoop tras la copia

6.2.12 Funcionamiento por defecto (copiado de archivos)

Por último, se muestran los resultados de la sincronización por defecto, es decir, sin especificar ningún tipo de opción: "*python rsyncHdfs.py -d hdfstoloc -pL <path en Linux de Universidad> -pH <path en Hadoop de Universidad>".*

Nombre del Directorio	***Universidad***	
Sincronización	**Antes**	**Después**
Subdirectorios en Hadoop	*Apuntes* *Examenes* *Notas* *Practicas* *Trabajos*	*Apuntes* *Examenes* *Notas* *Practicas* *Trabajos*
Subdirectorios en Linux	*Apuntes* *Examenes* *Notas* *Practicas* *Trabajos*	*Apuntes* *Examenes* *Notas* *Practicas* *Trabajos*
Ficheros en Hadoop	*Cosas_Varias_Uni.zip*	*Cosas_Varias_Uni.zip*
Ficheros en Linux	-	*Cosas_Varias_Uni.zip*

Tabla 6.2.12: Funcionamiento por defecto del sincronizador

Si se quiere observar el estado real de ambos directorios (origen y destino) tras la ejecución de cada una de las pruebas, los pantallazos con los resultados de las mismas se encuentran recogidos más adelante, dentro del Apéndice D.

6.3 Conclusiones obtenidas de las pruebas

La elaboración de este Trabajo Final de Grado ha tenido como resultado la herramienta RsyncHdfs, la cual cumple con los objetivos marcados al principio. Los diferentes tipos de pruebas realizadas han sido de utilidad para validar la versión final, hasta el momento, del software desarrollado.

No se ha producido ningún tipo de incidencia grave durante el desarrollo de las pruebas. RsyncHdfs ha demostrado ser robusto durante la ejecución de las mismas, llevándose a cabo todas ellas de manera satisfactoria. Además, es una herramienta fácil de utilizar, ya que cuenta con un manual de uso en el que se detalla cómo se ha de ejecutar cada una de las opciones disponibles para el usuario.

Por último, a la vista de los resultados obtenidos en las pruebas, se puede concluir que el software desarrollado permite ahorrar mucho tiempo de comprobaciones de ficheros cuando se necesita tener la información más actual para poder disponer de ella y utilizarla.

7 Conclusiones y Trabajo Futuro

7.1 Conclusiones

Para finalizar este trabajo, en esta sección se desarrollan las conclusiones generales y algunas ideas que podrían llevarse a cabo para mejorar el sincronizador.

Con la elaboración de RsyncHdfs se ha cumplido satisfactoriamente el objetivo de este Trabajo Final de Grado: crear un software que de soporte a la transferencia y actualización de ficheros y directorios entre sistemas Linux y Hadoop.

El sincronizador elaborado se ha convertido en una herramienta muy útil, permitiendo la sincronización de una gran cantidad de archivos en, o a partir de, Hadoop sin necesidad de ir revisando cada uno por separado.

El potencial alcanzado por la herramienta se ha manifestado en el transcurso de las pruebas realizadas. A lo largo de la ejecución de las mismas se ha podido ir comprobando el uso del sincronizador, quedando al descubierto de este mismo modo posibles mejoras que pueden llevarse a cabo para contar con una herramienta más potente y competitiva.

Por otra parte, desde el punto de vista personal, este trabajo me ha permitido poner en práctica los conocimientos adquiridos durante el grado sobre el entorno Linux, así como abordar un trabajo de principio a fin en el que he adquirido nuevos conocimientos sobre Hadoop, las tecnologías utilizadas en el entorno Big Data y aprender más sobre el funcionamiento en las sincronizaciones de archivos.

7.2 Trabajo futuro

Se han considerado las siguientes ideas como posible trabajo futuro:

1. Crear un controlador del sincronizador que le permita al usuario ejecutarlo para unas rutas específicas cada cierto tiempo sin necesidad de tener que ejecutarlo él mismo, para de esta forma actualizar automáticamente los archivos que desee.

2. Añadir un nuevo parámetro al sincronizador que permita al usuario poder mantener, si lo desea, en un mismo directorio ficheros con el mismo nombre pero fechas distintas, es decir, distintas versiones de un mismo archivo.

3. Añadir un nuevo parámetro al sincronizador que muestre al usuario el manual de uso, para de esta forma no tener que abrir directamente el fichero que lo contiene.

4. Ampliar la herramienta para que no solo actualice directorios, sino que a la vez que actualiza estos, actualice las tablas de Hive que utilicen alguno de los ficheros que contengan.

5. Realizar pruebas más exhaustivas en entornos de trabajo reales con una mayor cantidad de datos, para comprobar la eficiencia de la sincronización.

6. Comprobar con trabajadores del sector si realmente es fácil la utilización de la herramienta y asegurarse que su funcionamiento resulta cómodo para los usuarios.

Referencias

[1] *Introducción a Hadoop y su ecosistema* (por: J. Casanella - 02/04/2013):
http://www.ticout.com/blog/2013/04/02/introduccion-a-hadoop-y-su-ecosistema/

[2] *Mover datos en Hadoop* (por: J. Casanella - 10/04/2018):
http://www.ticout.com/blog/2013/04/10/mover-datos-en-hadoop/

[3] *Sistema de archivos distribuido de Hadoop, HDFS* (por: IBM):
https://www.ibm.com/support/knowledgecenter/es/SSPT3X_4.1.0/com.ibm.swg.im.infosphere.biginsights.product.doc/doc/c0057606.html

[4] *¿Cómo se relacionan Big Data y Hadoop?* (por: PowerData - 05/09/2013):
https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/328879/c-mo-se-relacionan-big-data-y-hadoop

[5] *Qué es Rsync y cómo usarlo para respaldos* (por: Jonathan Carrillo - 16/11/2015):
https://www.solvetic.com/tutoriales/article/2170-que-es-rsync-y-como-usarlo-para-respaldos/

[6] *Hadoop, la plataforma Open Source que lidera Big Data* (por: Pablo G. Bejerano - 09/08/2013):
https://blogthinkbig.com/hadoop-open-source-big-data

[7] *Origen, historia y situación actual de Linux* (por: Marilu Choque Cuevas - 11/08/2010):
http://lecturasdelavida.blogspot.com.es/p/origenhistoria-y-situacion-actual-de.html

[8] *Organización del sistema de archivos en Linux* (por: Luis Llamas - 14/10/2013):
https://www.luisllamas.es/organizacion-sistema-de-archivos-en-linux/

[9] *Administración Linux: Sistemas de Archivos* (por: Digital Learning - 11/03/2012):
http://www.digitallearning.es/blog/administracion-linux-sistemas-de-archivos/

[10] *El árbol de directorios de Linux al detalle. Principales carpetas y sus funciones* (por: Uri - 14/06/2015):
https://computernewage.com/2015/06/14/el-arbol-de-directorios-de-linux-al-detalle-que-contiene-cada-carpeta/

[11] *5 ventajas de la arquitectura de Hadoop* (por: PowerData - 16/02/2015):
https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/402826/5-ventajas-de-la-arquitectura-de-Hadoop

[12] *¿Qué es Cloudera?* (por: Bonisú Fernández García):
https://www.clarcat.com/cloudera

[13] *Big Data y Hadoop. Cloudera vs Hortonworks* (por: Gorka Hurtado - 30/01/2014):
https://mukom.mondragon.edu/ict/big-data-y-hadoop-cloudera-vs-hortonworks/

[14] *HDInsight: Big Data en el ecosistema de Microsoft* (por: esmsdn - 26/12/2012): Hadoop: https://blogs.msdn.microsoft.com/esmsdn/2012/12/26/hdinsight-big-data-en-el-ecosistema-microsoft/

[15] *Sincronización a fondo con Rsync en Ubuntu, Linux Mint y derivados* (por: Lorenzo Carbonell - 03/07/2017): https://www.atareao.es/software-linux/sincronizacion-a-fondo-con-rsync/

[16] *Rsync*: https://rsync.samba.org/ https://linux.die.net/man/1/rsync

[17] *Resilio Sync* (por: Stephanie De Puy - 01/10/2016): http://pontegeekpty.com/resilio-sync/

[18] *Resilio*: https://www.resilio.com/

[19] *Syncthing: interesante software para sincronizar archivos* (por: Tannhausser - 17/06/2015): https://lamiradadelreplicante.com/2015/06/17/syncthing-interesante-software-para-sincronizar-archivos/

[20] *Syncthing*: https://syncthing.net/

[21] *Nextcloud*: https://nextcloud.com/

[22] *Nextcloud* (por: Redes Zone): https://www.redeszone.net/nextcloud/

Glosario

- **HDFS:** Hadoop Distributed File System.
- **CD:** Compact Disc.
- **DVD:** Digital Versatile Disc.
- **FHS:** Filesystem Hierarchy Standard
- **GRUB:** GNU Rand Unified Bootloader
- **USB:** Universal Serial Bus
- **CDROM:** Compact Disc Read-Only Memory
- **GFS:** Google File System
- **CDH:** Cloudera Distribution Hadoop
- **GUI:** Graphical User Interface
- **GTK:** Gimp Tool Kit

Anexos

A Manual de instalación

Para poder hacer un uso correcto en local de la herramienta, se han de seguir una serie de pasos.

Lo primero de todo es necesario el uso de un sistema operativo de distribución Linux, ya que el sincronizador está pensado para sincronizar directorios de Linux con Hadoop y se trabaja con él a través de la terminal de Linux. Además, habrá que instalar Python v2.7.3 y cualquier versión de Cloudera para poder disponer de Hadoop.

Se puede instalar todo esto dentro de una máquina virtual en un sistema operativo de Windows, para poder trabajar sin necesidad de reinstalar el sistema de nuestro ordenador. Para poder utilizar la máquina virtual será necesario tener instalado VirtualBox o VMWare Workstation, la versión que se encuentre en estos momentos disponible para el software.

Por último, para ejecutar el sincronizador simplemente tenemos que ejecutar el comando *"python rsyncHdfs.py"* en una terminal de Linux, con los atributos necesarios para nuestra sincronización. Es importante señalar que para poder ejecutar la herramienta tendremos que situarnos en la ruta en la que se encuentre RsyncHdfs, ya sea comenzando la ejecución desde ahí o navegando a través de la terminal hasta dicho directorio.

B Manual de uso (README)

```
SYNCHRONIZER INSTRUCTIONS:

    - Use Example: rsyncHdfs.py -dir <string for indicating direction loc2Hdfs or Hdfs2loc> -cp <value Y or N> -rm <value Y or N> -u <value Y or
                               -pL <string with value for the path local> -pH <string with value for the path hadoop/hdfs> -rmOri <value Y or N

    - Attributes:

        > direction (-dir): "locToHdfs" for synchronizing files from Local to HDFS.
                            "hdfsToLoc" for synchronizing files from HDFS to Local.

            -> Use Examples:    rsyncHdfs.py -dir locToHdfs -cp Y -rm N -u Y -pL /home/sergio/sources -pH /user/sergio/test -rmOri N
                                rsyncHdfs.py -dir hdfsToLoc -cp Y -rm N -u Y -pL /home/sergio/sources -pH /user/sergio/test -rmOri N

        > copy (-cp):           "Y" copy. Don't update files/directories existing in the synchronization.

            -> Use Examples:    rsyncHdfs.py -dir locToHdfs -cp Y -rm N -u N -pL /home/sergio/sources -pH /user/sergio/test -rmOri N
                                rsyncHdfs.py -dir hdfsToLoc -cp Y -rm N -u N -pL /home/sergio/sources -pH /user/sergio/test -rmOri N

            >> deleteOrigin (-rmOri):   "Y" delete files in origin if files be copied in destination.
                                        "N" don't delete files in origin.

                ->> Use Examples:   rsyncHdfs.py -dir loc2Hdfs -cp Y -rm N -u N -pL /home/sergio/sources -pH /user/sergio/test -rmOri Y
                                    rsyncHdfs.py -dir Hdfs2loc -cp Y -rm N -u N -pathLocal /home/sergio/sources -pH /user/sergio/test -rmOri Y

        > delete (-rm):     "Y" delete files/directories from destination if they aren´t in source.
                            "N" don't delete files/directories.

            -> Use Examples:    rsyncHdfs.py -dir loc2Hdfs -cp Y -rm Y -u N -pL /home/sergio/sources -pH /user/sergio/test -rmOri N
                                rsyncHdfs.py -dir Hdfs2loc -cp Y -rm Y -u N -pL /home/sergio/sources -pH /user/sergio/test -rmOri N

        > update (-u):      "Y" update. Don't add new files/directories in the synchronization.

            -> Use Examples:    rsyncHdfs.py -dir loc2Hdfs -copy Y -rm Y -u Y -pL /home/sergio/sources -pH /user/sergio/test -rmOri N
                                rsyncHdfs.py -dir Hdfs2loc -copy N -rm N -u N -pL /home/sergio/sources -pH /user/sergio/test -rmOri N

        > pathLocal (-pL):  path with local directory. Ex: /home/sergio/sources
        > pathHadoop (-pH): path with hdfs directory. Ex: /user/sergio/test

    - Options:

        > If rm = Y: Delete files in destination if they aren´t in source.

        > If cp = Y: Copy files from source to destination if not exist in destination.
            >> If rmOri = Y: Delete files in origin if files be copied in destination.

        > If u = Y: Update files from source to destination if exist in destination and if date of destination is older than date of source.

        > If (cp = Y and u = Y): Copy and update files from source to destination if not exist or if exist in destination.

        > If (cp = Y and rm = Y): Copy files from source to destination if not exist in destination
                                  and delete files/directories from destination if they aren´t in source.

        > If (u = Y and rm = Y): Update files from source to destination if exist in destination and delete files/directories
                                 from destination if they aren´t in source.

        > If (cp = Y and u = Y and rm = Y): Copy and update files from source to destination if not exist or if exist in destination
                                           and delete files/directories from destination if they aren´t in source.
```

Figura B.1: Manual de RsyncHdfs

C Estructura de archivos de las pruebas

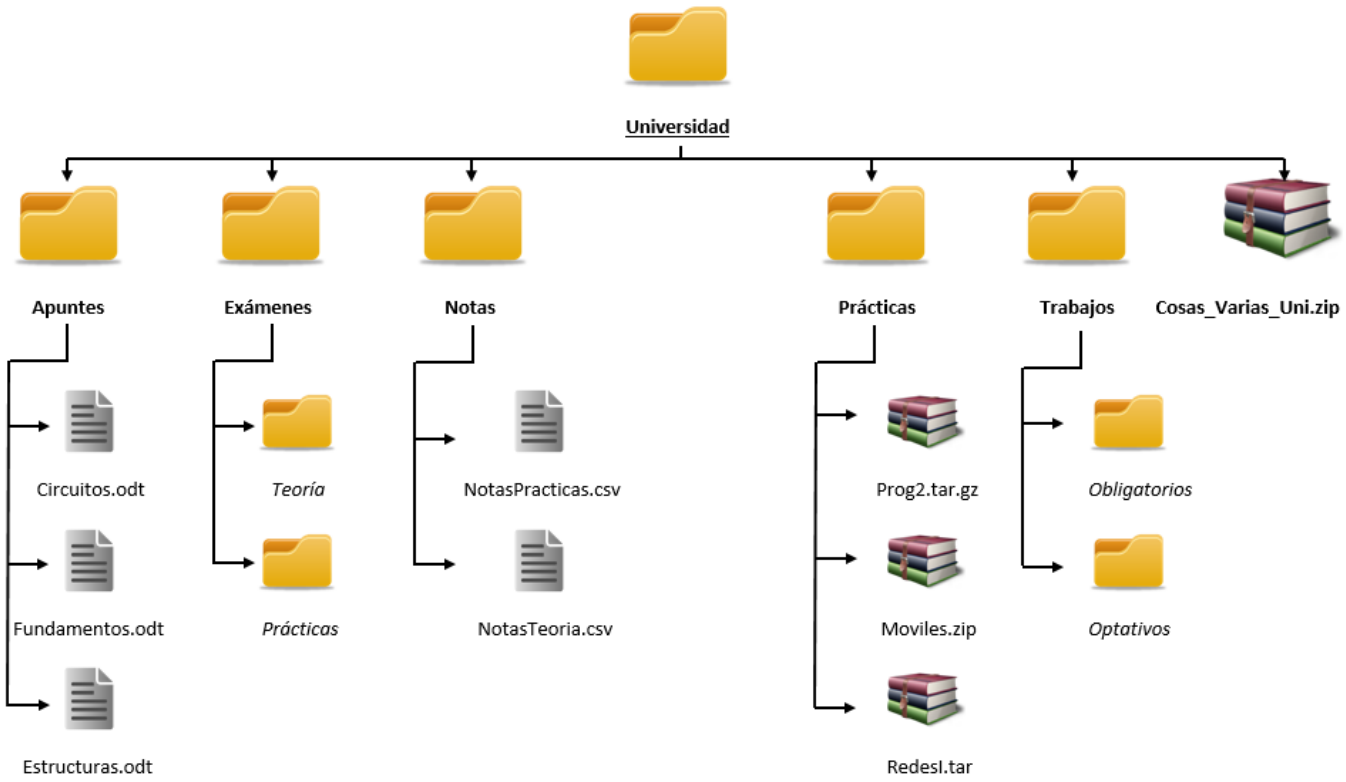

Figura C.1: Estructura de archivos de las pruebas

En la imagen etiquetada como “Figura C.1” de este anexo, puede observarse una toma parcial de la que ha sido la estructura de directorios y archivos utilizada para las pruebas redactadas en el apartado “7.2 Pruebas de usabilidad y resultados”.

A continuación, en el siguiente anexo se muestran las imágenes con los resultados reales obtenidos en las pruebas.

D Resultados de las pruebas

D.1 Sentido de la sincronización de Linux a Hadoop

D.1.1 Copiado de archivos existentes

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/Apuntes/
total 12
-rwxrwxrwx 1 cloudera cloudera 102 Apr 11 08:55 Circuitos.odt
-rwxrwxrwx 1 cloudera cloudera  64 Apr 11 08:56 Estructuras.odt
-rwxrwxrwx 1 cloudera cloudera  64 Apr 11 08:56 Fundamentos.odt
```

Figura D.1.1.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Apuntes
Found 2 items
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Estructuras.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Fundamentos.odt
```

Figura D.1.1.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d loctohdfs -pL /home/sergio/Universidad/Apuntes
-pH /test/data/sergio/Universidad/Apuntes -cp Y
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Apuntes/tempFolder_2018-04-11_09.25.26.670683_Apun
tes
[cloudera@quickstart sergio]$ ls -l Universidad/Apuntes/
total 12
-rwxrwxrwx 1 cloudera cloudera 102 Apr 11 08:55 Circuitos.odt
-rwxrwxrwx 1 cloudera cloudera  64 Apr 11 08:56 Estructuras.odt
-rwxrwxrwx 1 cloudera cloudera  64 Apr 11 08:56 Fundamentos.odt
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Apuntes
Found 3 items
-rwxrwxrwx   1 cloudera cloudera        102 2018-04-11 08:55 /test/data/sergio/Universidad/Apuntes/Circuitos.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Estructuras.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Fundamentos.odt
```

Figura D.1.1.3: Copiado y estado final de los directorios

D.1.2 Actualización de archivos

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/Notas/
total 24
-rwxrwxrwx+ 1 sergio sergio 11995 Apr 11 09:44 NotasPracticas.xlsx
-rwxrwxrwx+ 1 sergio sergio 10226 Apr 11 09:43 NotasTeoria.xlsx
```

Figura D.1.2.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Notas
Found 2 items
-rwxrwxrwx   1 cloudera cloudera      10156 2018-04-11 08:53 /test/data/sergio/Universidad/Notas/NotasPracticas.xlsx
-rwxrwxrwx   1 cloudera cloudera      10158 2018-04-11 08:53 /test/data/sergio/Universidad/Notas/NotasTeoria.xlsx
```

Figura D.1.2.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d loctohdfs -pL /home/sergio/Universidad/Notas -pH /test/
data/sergio/Universidad/Notas -u Y
copyFromLocal: Non-super user cannot change owner
copyFromLocal: Non-super user cannot change owner
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Notas
Found 2 items
-rw-r--r--   1 cloudera cloudera      11995 2018-04-11 09:44 /test/data/sergio/Universidad/Notas/NotasPracticas.xlsx
-rw-r--r--   1 cloudera cloudera      10226 2018-04-11 09:43 /test/data/sergio/Universidad/Notas/NotasTeoria.xlsx
[cloudera@quickstart sergio]$ ls -l Universidad/Notas/
total 24
-rwxrwxrwx+ 1 sergio sergio 11995 Apr 11 09:44 NotasPracticas.xlsx
-rwxrwxrwx+ 1 sergio sergio 10226 Apr 11 09:43 NotasTeoria.xlsx
```

Figura D.1.2.3: Actualización y estado final de los directorios

D.1.3 Copiado y actualización de archivos existentes

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/
total 8
drwxrwxrwx 2 sergio sergio 4096 Apr 11 09:30 Practicas
drwxrwxrwx 2 sergio sergio 4096 Apr 11 08:39 Teoria
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Practicas/
total 12
-rwxr--r-- 1 sergio sergio 1528 Apr 11 08:40 Pautlen.zip
-rwxr--r-- 1 sergio sergio 1693 Apr 11 09:30 PSI.zip
-rwxr--r-- 1 sergio sergio 1606 Apr 11 08:40 Soper.zip
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Teoria/
total 8
-rwxr--r-- 1 sergio sergio 1528 Apr 11 08:37 Redes1.zip
-rwxr--r-- 1 sergio sergio 1528 Apr 11 08:38 Redes2.zip
```

Figura D.1.3.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes
Found 1 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 08:41 /test/data/sergio/Universidad/Examenes/Practicas
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Practicas
Found 3 items
-rwxr--r--   1 cloudera cloudera       1696 2018-04-11 08:41 /test/data/sergio/Universidad/Examenes/Practicas/PSI.zip
-rwxr--r--   1 cloudera cloudera       1528 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Pautlen.zip
-rwxr--r--   1 cloudera cloudera       1606 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Soper.zip
```

Figura D.1.3.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d loctohdfs -pL /home/sergio/Universidad/Examenes -pH /t
st/data/sergio/Universidad/Examenes -cp Y -u Y
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Practicas
Found 3 items
-rw-r--r--   1 cloudera cloudera       1693 2018-04-11 09:30 /test/data/sergio/Universidad/Examenes/Practicas/PSI.zip
-rwxr--r--   1 cloudera cloudera       1528 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Pautlen.zip
-rwxr--r--   1 cloudera cloudera       1606 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Soper.zip
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes
Found 2 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:36 /test/data/sergio/Universidad/Examenes/Practicas
drwxr-xr-x   - cloudera cloudera          0 2018-04-11 08:39 /test/data/sergio/Universidad/Examenes/Teoria
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Teoria
Found 2 items
-rw-r--r--   1 cloudera cloudera       1528 2018-04-11 08:37 /test/data/sergio/Universidad/Examenes/Teoria/Redes1.zip
-rw-r--r--   1 cloudera cloudera       1528 2018-04-11 08:38 /test/data/sergio/Universidad/Examenes/Teoria/Redes2.zip
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/
total 8
drwxrwxrwx 2 sergio sergio 4096 Apr 11 09:30 Practicas
drwxrwxrwx 2 sergio sergio 4096 Apr 11 08:39 Teoria
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Practicas/
total 12
-rwxr--r-- 1 sergio sergio 1528 Apr 11 08:40 Pautlen.zip
-rwxr--r-- 1 sergio sergio 1693 Apr 11 09:30 PSI.zip
-rwxr--r-- 1 sergio sergio 1606 Apr 11 08:40 Soper.zip
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Teoria/
total 8
-rwxr--r-- 1 sergio sergio 1528 Apr 11 08:37 Redes1.zip
-rwxr--r-- 1 sergio sergio 1528 Apr 11 08:38 Redes2.zip
```

Figura D.1.3.3: Copia, actualización y estado final de los directorios

D.1.4 Copiado y eliminación de archivos existentes en Hadoop

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/Practicas/
total 16
-rwxr--r-- 1 sergio sergio 514 Apr  9 09:23 Moviles.zip
-rwxr--r-- 1 sergio sergio 536 Apr  9 09:23 Prog2.tar.gz
-rwxr--r-- 1 sergio sergio 416 Apr  9 09:23 RedesI.tar
-rwxr--r-- 1 sergio sergio 514 Apr  9 09:23 Videojuegos.zip
```

Figura D.1.4.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Practicas
Found 5 items
-rwxr--r--   1 cloudera cloudera        536 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Moviles_Viejo.gz
-rwxr--r--   1 cloudera cloudera        536 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Prog2.tar.gz
-rwxr--r--   1 cloudera cloudera        416 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/PruebasMoviles.tar
-rwxr--r--   1 cloudera cloudera        416 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/RedesI.tar
-rwxr--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Videojuegos.zip
```

Figura D.1.4.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d loctohdfs -pL /home/sergio/Universidad/Practicas -pH /t
est/data/sergio/Universidad/Practicas -cp Y -rm Y
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Practicas/Moviles_Viejo.gz
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Practicas/PruebasMoviles.tar
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Practicas/tempFolder_2018-04-11_09.57.57.836472_Practicas
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Practicas
Found 4 items
-rwxr--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Moviles.zip
-rwxr--r--   1 cloudera cloudera        536 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Prog2.tar.gz
-rwxr--r--   1 cloudera cloudera        416 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/RedesI.tar
-rwxr--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Videojuegos.zip
[cloudera@quickstart sergio]$ ls -l Universidad/Practicas/
total 16
-rwxr--r-- 1 sergio sergio 514 Apr  9 09:23 Moviles.zip
-rwxr--r-- 1 sergio sergio 536 Apr  9 09:23 Prog2.tar.gz
-rwxr--r-- 1 sergio sergio 416 Apr  9 09:23 RedesI.tar
-rwxr--r-- 1 sergio sergio 514 Apr  9 09:23 Videojuegos.zip
```

Figura D.1.4.3: Copia, eliminación y estado final de los directorios

D.1.5 Copiado, eliminación de archivos existentes en Hadoop y eliminación de los copiados

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/
total 8
drwxrwxrwx 2 sergio sergio 4096 Apr 11 10:03 Obligatorios
drwxrwxrwx 2 sergio sergio 4096 Apr  9 09:20 Optativos
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/Obligatorios/
total 12
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:22 Algebra.odt
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:22 ElectroMagnetismo.odt
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:22 PINGS.odt
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/Optativos/
total 8
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:21 InteligenciaArtificial.odt
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:21 ProbabilidadyEstadisticaEmpresarial.odt
```

Figura D.1.5.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos
Found 1 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:03 /test/data/sergio/Universidad/Trabajos/Obligatorios
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos/Obligatorios
Found 5 items
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/Algebra.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/Algebra_cop
ia.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/Calculo.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/ElectroMagn
etismo.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/PINGS.odt
```

Figura D.1.5.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d loctohdfs -pL /home/sergio/Universidad/Trabajos -pH /test/
data/sergio/Universidad/Trabajos -cp Y -rm Y -rmOri Y
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Trabajos/Obligatorios/Algebra_copia.odt
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Trabajos/Obligatorios/Calculo.odt
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Trabajos/tempFolder_2018-04-11_10.08.15.200861_Trabajos
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos
Found 2 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:08 /test/data/sergio/Universidad/Trabajos/Obligatorios
drwxrwxr--   - cloudera cloudera          0 2018-04-09 09:20 /test/data/sergio/Universidad/Trabajos/Optativos
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos/Obligatorios
Found 3 items
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/Algebra.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/ElectroMagnet
ismo.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/PINGS.odt
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos/Optativos
Found 2 items
-rw-r--r--   1 cloudera cloudera         49 2018-04-09 09:21 /test/data/sergio/Universidad/Trabajos/Optativos/InteligenciaArti
ficial.odt
-rw-r--r--   1 cloudera cloudera         49 2018-04-09 09:21 /test/data/sergio/Universidad/Trabajos/Optativos/ProbabilidadyEst
adisticaEmpresarial.odt
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/
total 4
drwxrwxrwx 2 sergio sergio 4096 Apr 11 10:03 Obligatorios
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/Obligatorios/
total 12
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:22 Algebra.odt
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:22 ElectroMagnetismo.odt
-rwxr--r-- 1 sergio sergio 49 Apr  9 09:22 PINGS.odt
```

Figura D.1.5.3: Copia, eliminación en origen y destino, y estado final de los directorios

D.1.6 Copiado, actualización y eliminación de los copiados o actualizados

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/
total 24
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 08:35 Apuntes
-rwxrwxrwx 1 sergio   sergio    278 Apr  9 09:13 Cosas_Varias_Uni.zip
drwxrwxrwx 4 cloudera cloudera 4096 Apr  9 09:11 Examenes
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 09:44 Notas
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 09:54 Practicas
drwxrwxrwx 3 sergio   sergio   4096 Apr 11 10:08 Trabajos
```

Figura D.1.6.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad
Found 5 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:25 /test/data/sergio/Universidad/Apuntes
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:37 /test/data/sergio/Universidad/Examenes
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:47 /test/data/sergio/Universidad/Notas
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:58 /test/data/sergio/Universidad/Practicas
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:08 /test/data/sergio/Universidad/Trabajos
```

Figura D.1.5.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d loctohdfs -pL /home/sergio/Universidad -pH /test/data/serg
io/Universidad -cp Y -u Y -rmOri Y
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad
Found 6 items
drwxrwxrwx   - cloudera cloudera            0 2018-04-11 09:25 /test/data/sergio/Universidad/Apuntes
-rw-r--r--   1 cloudera supergroup        278 2018-04-09 09:13 /test/data/sergio/Universidad/Cosas_Varias_Uni.zip
drwxrwxrwx   - cloudera cloudera            0 2018-04-11 09:37 /test/data/sergio/Universidad/Examenes
drwxrwxrwx   - cloudera cloudera            0 2018-04-11 10:16 /test/data/sergio/Universidad/Notas
drwxrwxrwx   - cloudera cloudera            0 2018-04-11 10:17 /test/data/sergio/Universidad/Practicas
drwxrwxrwx   - cloudera cloudera            0 2018-04-11 10:08 /test/data/sergio/Universidad/Trabajos
[cloudera@quickstart sergio]$ ls -l Universidad/
total 0
```

Figura D.1.5.3: Copia, eliminación en origen y destino, y estado final de los directorios

D.2 Sentido de la sincronización de Hadoop a Linux

D.2.1 Copiado de archivos existentes

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Apuntes
Found 3 items
-rwxrwxrwx   1 cloudera cloudera        102 2018-04-11 08:55 /test/data/sergio/Universidad/Apuntes/Circuitos.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Estructuras.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Fundamentos.odt
```

Figura D.2.1.1: Estado inicial del directorio en Hadoop

```
[cloudera@quickstart sergio]$ ls -l Universidad/Apuntes/
total 4
-rwxrwxrwx 1 cloudera cloudera 102 Apr 11 08:55 Circuitos.odt
```

Figura D.2.1.2: Estado inicial del directorio en Linux

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d hdfsToloc -pL /home/sergio/Universidad/Apuntes
-pH /test/data/sergio/Universidad/Apuntes -cp Y
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Apuntes
Found 3 items
-rwxrwxrwx   1 cloudera cloudera        102 2018-04-11 08:55 /test/data/sergio/Universidad/Apuntes/Circuitos.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Estructuras.odt
-rwxrwxrwx   1 cloudera cloudera         64 2018-04-11 08:56 /test/data/sergio/Universidad/Apuntes/Fundamentos.odt
[cloudera@quickstart sergio]$ ls -l Universidad/Apuntes/
total 12
-rwxrwxrwx 1 cloudera cloudera 102 Apr 11 08:55 Circuitos.odt
-rwxrwxrwx 1 cloudera cloudera  64 Apr 11 08:56 Estructuras.odt
-rwxrwxrwx 1 cloudera cloudera  64 Apr 11 08:56 Fundamentos.odt
```

Figura D.2.1.3: Copiado y estado final de los directorios

D.2.2 NO Actualización de archivos

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Notas
Found 2 items
-rw-r--r--   1 cloudera cloudera      11995 2018-04-11 09:44 /test/data/sergio/Universidad/Notas/NotasPracticas.xlsx
-rw-r--r--   1 cloudera cloudera      10226 2018-04-11 09:43 /test/data/sergio/Universidad/Notas/NotasTeoria.xlsx
```

Figura D.2.2.1: Estado inicial del directorio en Hadoop

```
[cloudera@quickstart sergio]$ ls -l Universidad/Notas
total 28
-rwxr--r-- 1 sergio sergio 10049 Apr 16 09:17 NotasPracticas.xlsx
-rwxr--r-- 1 sergio sergio 12391 Apr 16 09:18 NotasTeoria.xlsx
```

Figura D.2.2.2: Estado inicial del directorio en Linux

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d hdfsToloc -pL /home/sergio/Universidad/Notas
-pH /test/data/sergio/Universidad/Notas -u Y
[cloudera@quickstart sergio]$ ls -l Universidad/Notas
total 28
-rwxr--r-- 1 sergio sergio 10049 Apr 16 09:17 NotasPracticas.xlsx
-rwxr--r-- 1 sergio sergio 12391 Apr 16 09:18 NotasTeoria.xlsx
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Notas
Found 2 items
-rw-r--r--   1 cloudera cloudera      11995 2018-04-11 09:44 /test/data/sergio/Universidad/Notas/NotasPracticas.xlsx
-rw-r--r--   1 cloudera cloudera      10226 2018-04-11 09:43 /test/data/sergio/Universidad/Notas/NotasTeoria.xlsx
```

Figura D.2.2.3: Sincronización finalizada y estado final de los directorios

D.2.3 Copiado y actualización de archivos existentes

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes
Found 2 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:36 /test/data/sergio/Universidad/Examenes/Practicas
drwxr-xr-x   - cloudera cloudera          0 2018-04-11 08:39 /test/data/sergio/Universidad/Examenes/Teoria
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Practicas
Found 3 items
-rw-r--r--   1 cloudera cloudera       1693 2018-04-11 09:30 /test/data/sergio/Universidad/Examenes/Practicas/PSI.zip
-rwxr--r--   1 cloudera cloudera       1528 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Pautlen.
ip
-rwxr--r--   1 cloudera cloudera       1606 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Soper.zi
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Teoria
Found 2 items
-rw-r--r--   1 cloudera cloudera       1528 2018-04-11 08:37 /test/data/sergio/Universidad/Examenes/Teoria/Redes1.zip
-rw-r--r--   1 cloudera cloudera       1528 2018-04-11 08:38 /test/data/sergio/Universidad/Examenes/Teoria/Redes2.zip
```

Figura D.2.3.1: Estado inicial del directorio en Hadoop

```
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/
total 4
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 08:39 Teoria
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Teoria/
total 8
-rwxrwxrwx 1 cloudera cloudera 1528 Apr 11 08:37 Redes1.zip
-rwxrwxrwx 1 cloudera cloudera 1528 Apr 11 08:38 Redes2.zip
```

Figura D.2.3.2: Estado inicial del directorio en Linux

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d hdfsToloc -pL /home/sergio/Universidad/Examenes
-pH /test/data/sergio/Universidad/Examenes -cp Y -u Y
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes
Found 2 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:36 /test/data/sergio/Universidad/Examenes/Practicas
drwxr-xr-x   - cloudera cloudera          0 2018-04-11 08:39 /test/data/sergio/Universidad/Examenes/Teoria
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Practicas
Found 3 items
-rw-r--r--   1 cloudera cloudera       1693 2018-04-11 09:30 /test/data/sergio/Universidad/Examenes/Practicas/PSI.z
p
-rwxr--r--   1 cloudera cloudera       1528 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Pautl
n.zip
-rwxr--r--   1 cloudera cloudera       1606 2018-04-11 08:40 /test/data/sergio/Universidad/Examenes/Practicas/Soper
zip
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Examenes/Teoria
Found 2 items
-rw-r--r--   1 cloudera cloudera       1528 2018-04-11 08:37 /test/data/sergio/Universidad/Examenes/Teoria/Redes1.z
p
-rw-r--r--   1 cloudera cloudera       1528 2018-04-11 08:38 /test/data/sergio/Universidad/Examenes/Teoria/Redes2.z
p
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/
total 8
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 09:36 Practicas
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 08:39 Teoria
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Practicas/
total 12
-rwxr--r-- 1 cloudera cloudera 1528 Apr 11 08:40 Pautlen.zip
-rw-r--r-- 1 cloudera cloudera 1693 Apr 11 09:30 PSI.zip
-rwxr--r-- 1 cloudera cloudera 1606 Apr 11 08:40 Soper.zip
[cloudera@quickstart sergio]$ ls -l Universidad/Examenes/Teoria/
total 8
-rwxrwxrwx 1 cloudera cloudera 1528 Apr 11 08:37 Redes1.zip
-rwxrwxrwx 1 cloudera cloudera 1528 Apr 11 08:38 Redes2.zip
```

Figura D.2.3.3: Sincronización finalizada y estado final de los directorios

D.2.4 Copiado y eliminación de archivos existentes en Linux

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Practicas
Found 4 items
-rw-r--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Moviles.zip
-rw-r--r--   1 cloudera cloudera        536 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Prog2.tar.gz
-rw-r--r--   1 cloudera cloudera        416 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/RedesI.tar
-rw-r--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Videojuegos.zi
```

Figura D.2.4.1: Estado inicial del directorio en Hadoop

```
[cloudera@quickstart sergio]$ ls -l Universidad/Practicas/
total 24
-rwxrwxrwx 1 cloudera cloudera 514 Apr  9 09:23 Moviles.zip
-rwxrwxrwx 1 cloudera cloudera 536 Apr  9 09:23 Prog2.tar.gz
-rwxr--r-- 1 sergio   sergio   416 Apr  9 09:23 RedesI_antiguo.tar
-rwxrwxrwx 1 cloudera cloudera 416 Apr  9 09:23 RedesI.tar
-rwxrwxrwx 1 cloudera cloudera 514 Apr  9 09:23 Videojuegos.zip
-rwxr--r-- 1 sergio   sergio   514 Apr  9 09:23 Videojuejos_antiguo.zip
```

Figura D.2.4.2: Estado inicial del directorio en Linux

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d hdfsToloc -pL /home/sergio/Universidad/Practica
 -pH /test/data/sergio/Universidad/Practicas -cp Y -rm Y
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Practicas
Found 4 items
-rw-r--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Moviles.zip
-rw-r--r--   1 cloudera cloudera        536 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Prog2.tar.gz
-rw-r--r--   1 cloudera cloudera        416 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/RedesI.tar
-rw-r--r--   1 cloudera cloudera        514 2018-04-09 09:23 /test/data/sergio/Universidad/Practicas/Videojuegos.zi
[cloudera@quickstart sergio]$ ls -l Universidad/Practicas/
total 16
-rwxrwxrwx 1 cloudera cloudera 514 Apr  9 09:23 Moviles.zip
-rwxrwxrwx 1 cloudera cloudera 536 Apr  9 09:23 Prog2.tar.gz
-rwxrwxrwx 1 cloudera cloudera 416 Apr  9 09:23 RedesI.tar
-rwxrwxrwx 1 cloudera cloudera 514 Apr  9 09:23 Videojuegos.zip
```

Figura D.2.4.3: Sincronización finalizada y estado final de los directorios

D.2.5 Copiado, eliminación de archivos existentes en Linux y eliminación de los copiados

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/
total 4
drwxrwxrwx 2 cloudera cloudera 4096 Apr  9 09:20 Optativos
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/Optativos/
total 8
-rwxrwxrwx 1 cloudera cloudera 49 Apr  9 09:21 InteligenciaArtificial.odt
-rwxrwxrwx 1 cloudera cloudera 49 Apr  9 09:21 ProbabilidadyEstadísticaEmpresarial.odt
```

Figura D.2.5.1: Estado inicial del directorio en Linux

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos
Found 2 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:08 /test/data/sergio/Universidad/Trabajos/Obligatorios
drwxrwxr--   - cloudera cloudera          0 2018-04-09 09:20 /test/data/sergio/Universidad/Trabajos/Optativos
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos/Obligatorios
Found 3 items
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/Al
ebra.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/El
ctroMagnetismo.odt
-rwxr--r--   1 cloudera cloudera         49 2018-04-09 09:22 /test/data/sergio/Universidad/Trabajos/Obligatorios/PI
GS.odt
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos/Optativos
Found 2 items
-rw-r--r--   1 cloudera cloudera         49 2018-04-09 09:21 /test/data/sergio/Universidad/Trabajos/Optativos/Intel
genciaArtificial.odt
-rw-r--r--   1 cloudera cloudera         49 2018-04-09 09:21 /test/data/sergio/Universidad/Trabajos/Optativos/Proba
ilidadyEstadísticaEmpresarial.odt
```

Figura D.2.5.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d hdfsToloc -pL /home/sergio/Universidad/Trabajos
-pH /test/data/sergio/Universidad/Trabajos -cp Y -rm Y -rmOri Y
Deleted hdfs://quickstart.cloudera:/test/data/sergio/Universidad/Trabajos/Obligatorios
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos
Found 1 items
drwxrwxr--   - cloudera cloudera          0 2018-04-16 09:58 /test/data/sergio/Universidad/Trabajos/Optativos
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad/Trabajos/Optativos
Found 1 items
-rw-r--r--   1 cloudera cloudera         49 2018-04-09 09:21 /test/data/sergio/Universidad/Trabajos/Optativos/Inteligen
iaArtificial.odt
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/
total 8
drwxrwxrwx 2 cloudera cloudera 4096 Apr 11 10:08 Obligatorios
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:57 Optativos
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/Obligatorios/
total 12
-rwxr--r-- 1 cloudera cloudera 49 Apr  9 09:22 Algebra.odt
-rwxr--r-- 1 cloudera cloudera 49 Apr  9 09:22 ElectroMagnetismo.odt
-rwxr--r-- 1 cloudera cloudera 49 Apr  9 09:22 PINGS.odt
[cloudera@quickstart sergio]$ ls -l Universidad/Trabajos/Optativos/
total 8
-rwxrwxrwx 1 cloudera cloudera 49 Apr  9 09:21 InteligenciaArtificial.odt
-rw-r--r-- 1 cloudera cloudera 49 Apr  9 09:21 ProbabilidadyEstadísticaEmpresarial.odt
```

Figura D.2.5.3: Copia, eliminación en origen y destino, y estado final de los directorios

D.2.6 Copiado, funcionamiento por defecto sin opciones indicadas

- **Directorios antes de la sincronización:**

```
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad
Found 6 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:25 /test/data/sergio/Universidad/Apuntes
-rw-r--r--   1 cloudera supergroup        278 2018-04-09 09:13 /test/data/sergio/Universidad/Cosas_Varias_
ni.zip
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:37 /test/data/sergio/Universidad/Examenes
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:16 /test/data/sergio/Universidad/Notas
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:17 /test/data/sergio/Universidad/Practicas
drwxrwxrwx   - cloudera cloudera          0 2018-04-16 09:57 /test/data/sergio/Universidad/Trabajos
```

Figura D.2.6.1: Estado inicial del directorio en Hadoop

```
[cloudera@quickstart sergio]$ ls -l Universidad/
total 20
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:27 Apuntes
drwxrwxrwx 4 cloudera cloudera 4096 Apr 16 09:39 Examenes
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:18 Notas
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:51 Practicas
drwxrwxrwx 4 cloudera cloudera 4096 Apr 16 09:57 Trabajos
```

Figura D.2.5.2: Estado inicial del directorio en Hadoop

- **Ejecución de la sincronización y directorios finales:**

```
[cloudera@quickstart sergio]$ python2.7 Desktop/TFG/rsyncHdfs.py -d hdfsToloc -pL /home/sergio/Universidad
-pH /test/data/sergio/Universidad
copyToLocal: chown: invalid group: `cloudera:supergroup'
[cloudera@quickstart sergio]$ ls -l Universidad/
total 24
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:27 Apuntes
-rwxrwxrwx 1 cloudera cloudera  278 Apr  9 09:13 Cosas_Varias_Uni.zip
drwxrwxrwx 4 cloudera cloudera 4096 Apr 16 09:39 Examenes
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:18 Notas
drwxrwxrwx 2 cloudera cloudera 4096 Apr 16 09:51 Practicas
drwxrwxrwx 3 cloudera cloudera 4096 Apr 16 10:07 Trabajos
[cloudera@quickstart sergio]$ hadoop fs -ls /test/data/sergio/Universidad
Found 6 items
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:25 /test/data/sergio/Universidad/Apuntes
-rw-r--r--   1 cloudera supergroup        278 2018-04-09 09:13 /test/data/sergio/Universidad/Cosas_Varias_
ni.zip
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 09:37 /test/data/sergio/Universidad/Examenes
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:16 /test/data/sergio/Universidad/Notas
drwxrwxrwx   - cloudera cloudera          0 2018-04-11 10:17 /test/data/sergio/Universidad/Practicas
drwxrwxrwx   - cloudera cloudera          0 2018-04-16 09:57 /test/data/sergio/Universidad/Trabajos
```

Figura D.1.5.3: Copia, eliminación en origen y destino, y estado final de los directorios